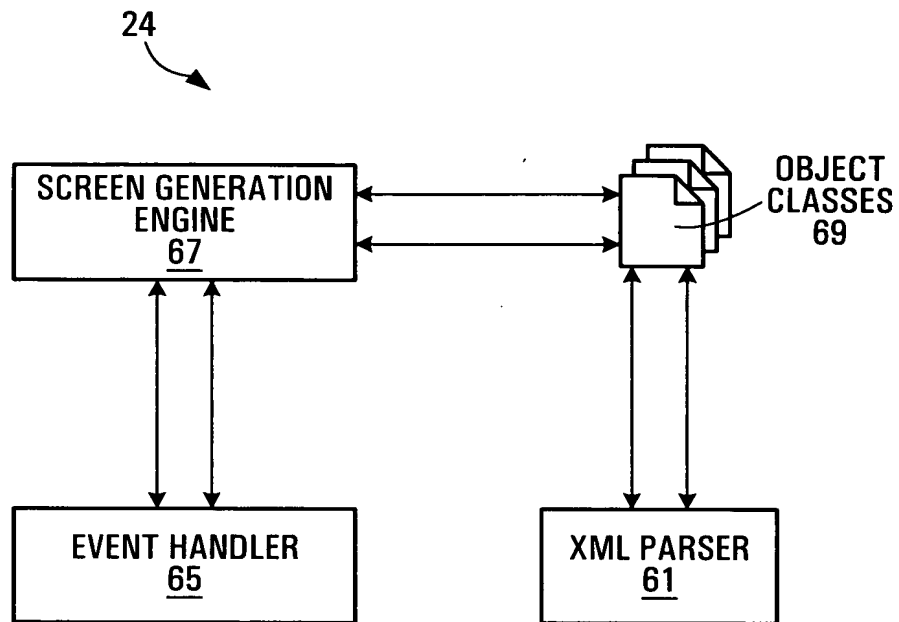


FIG. 1

**FIG. 2**

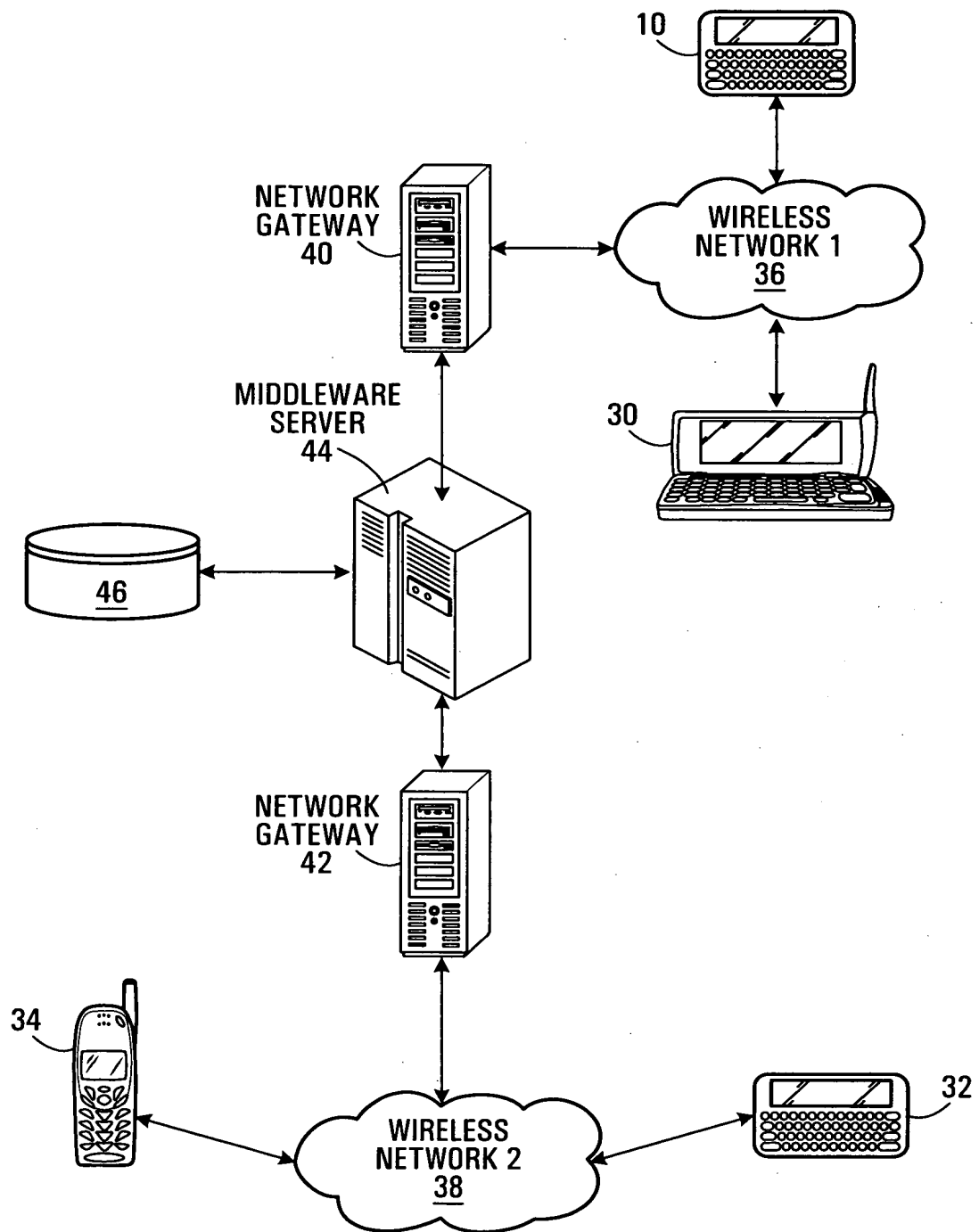


FIG. 3

4/84

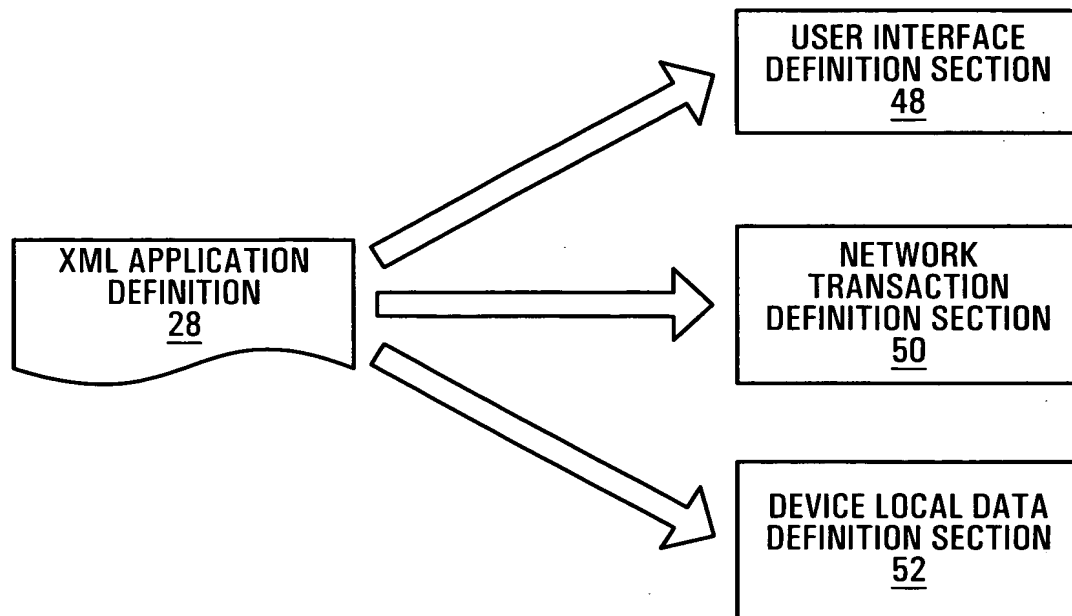


FIG. 4

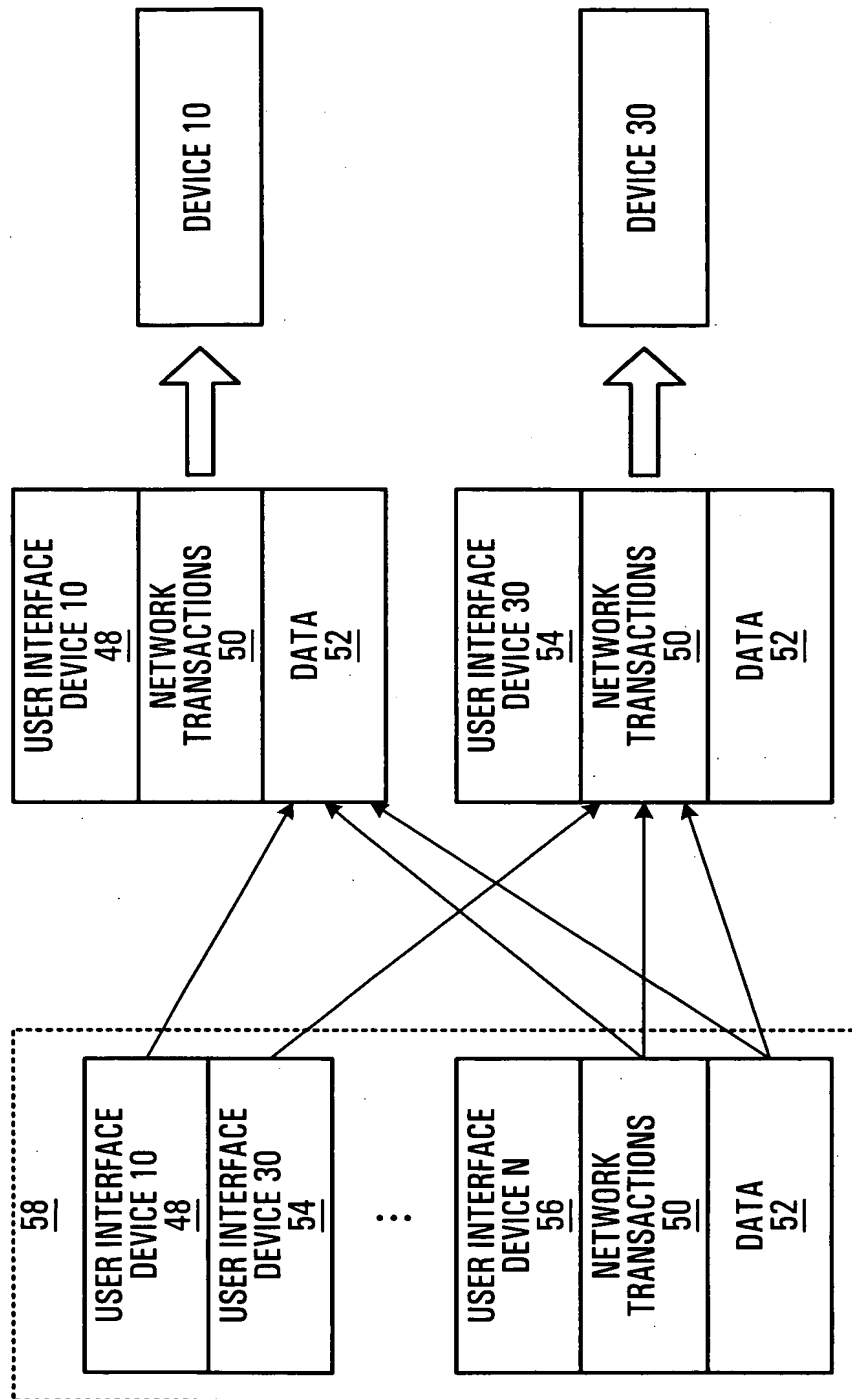


FIG. 5

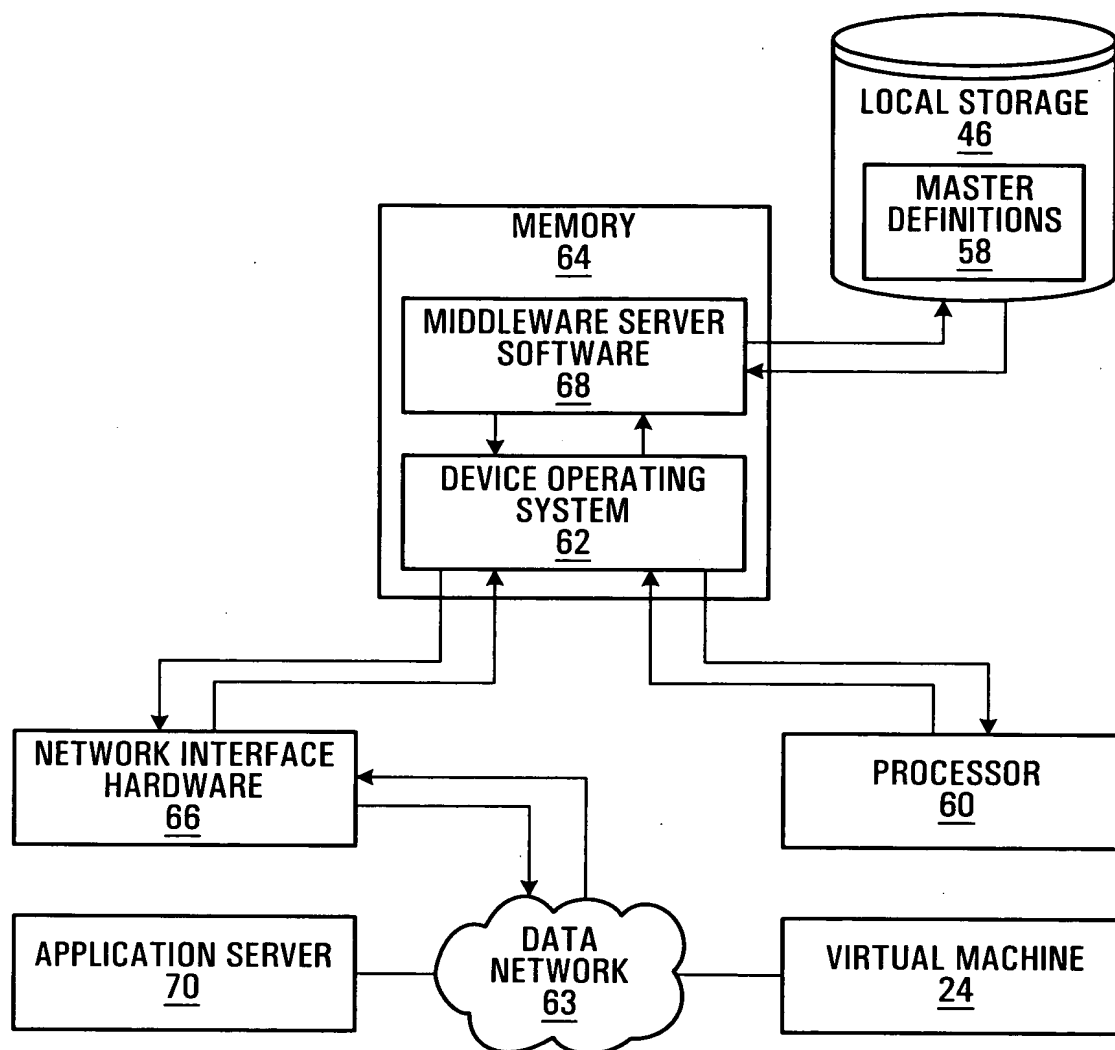


FIG. 6

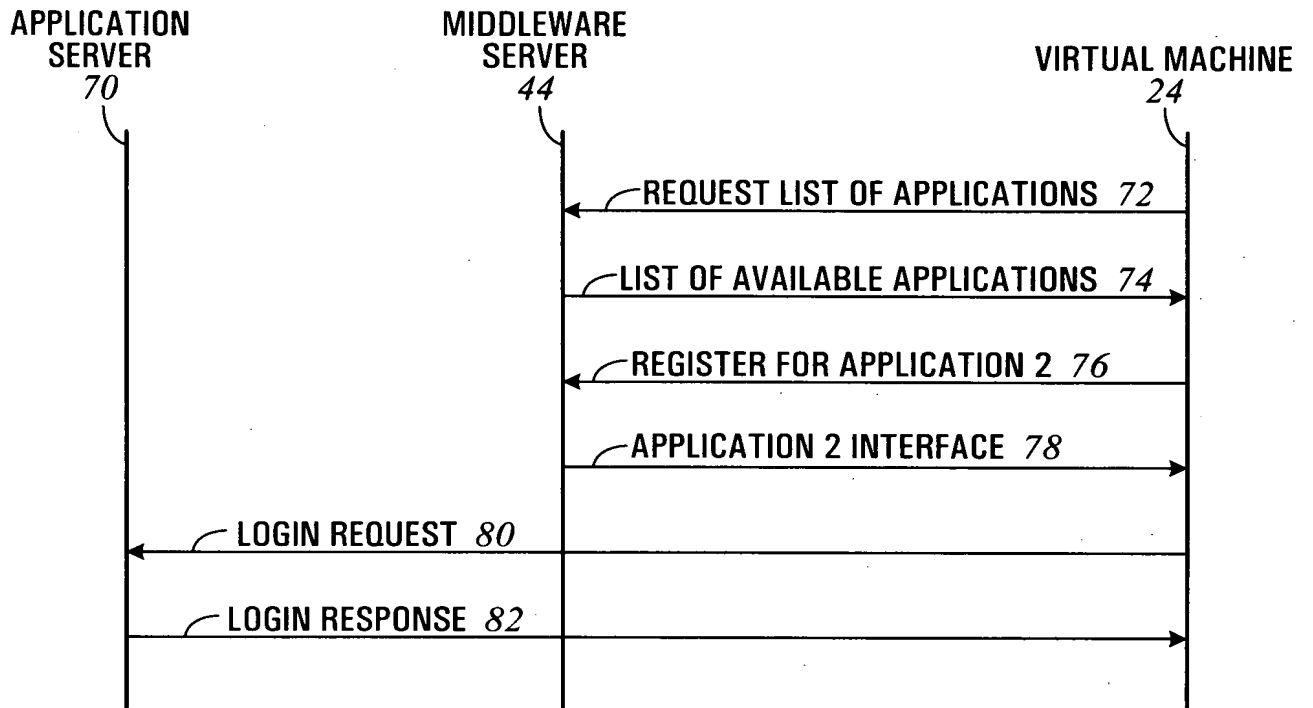


FIG. 7

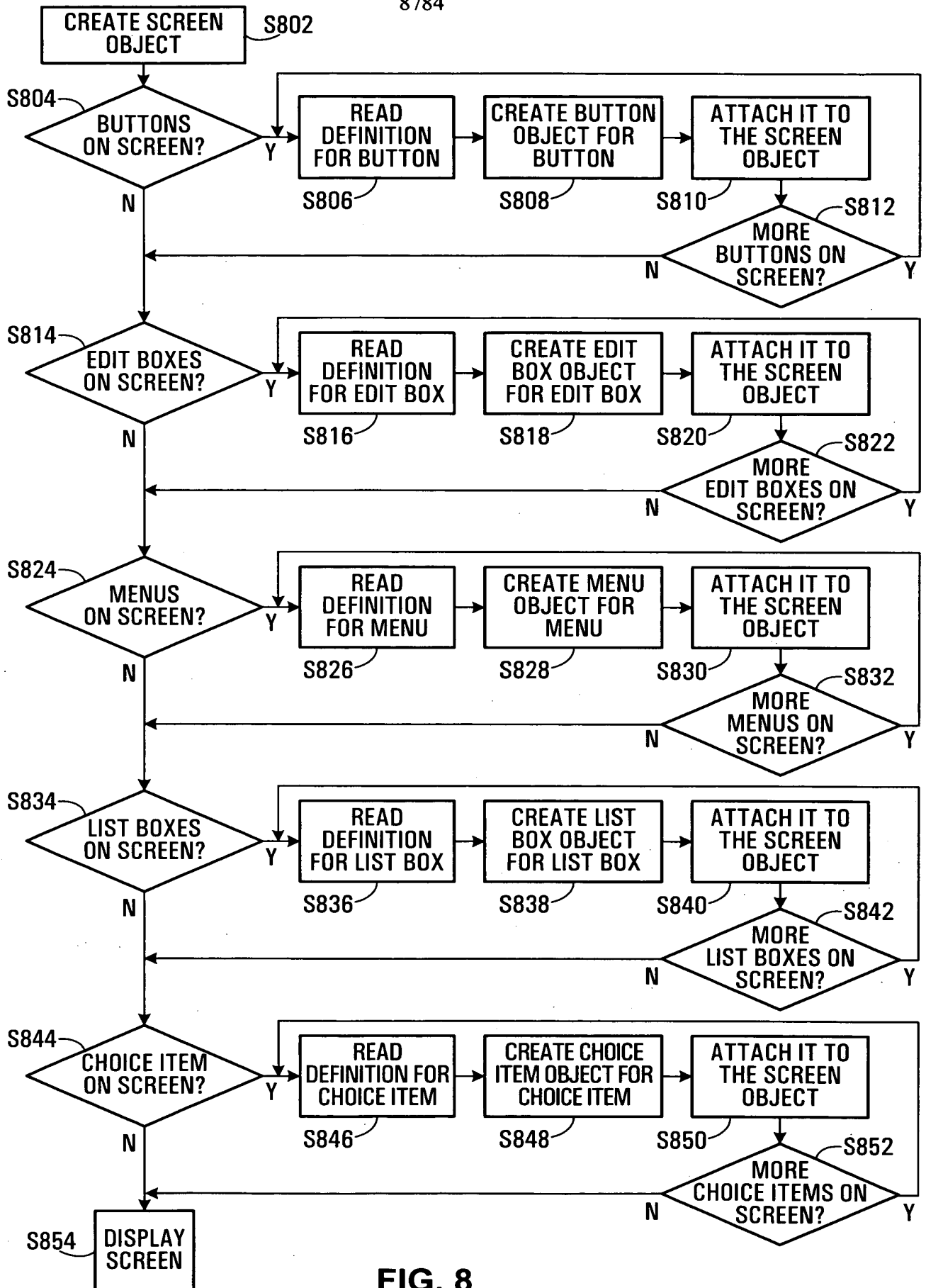
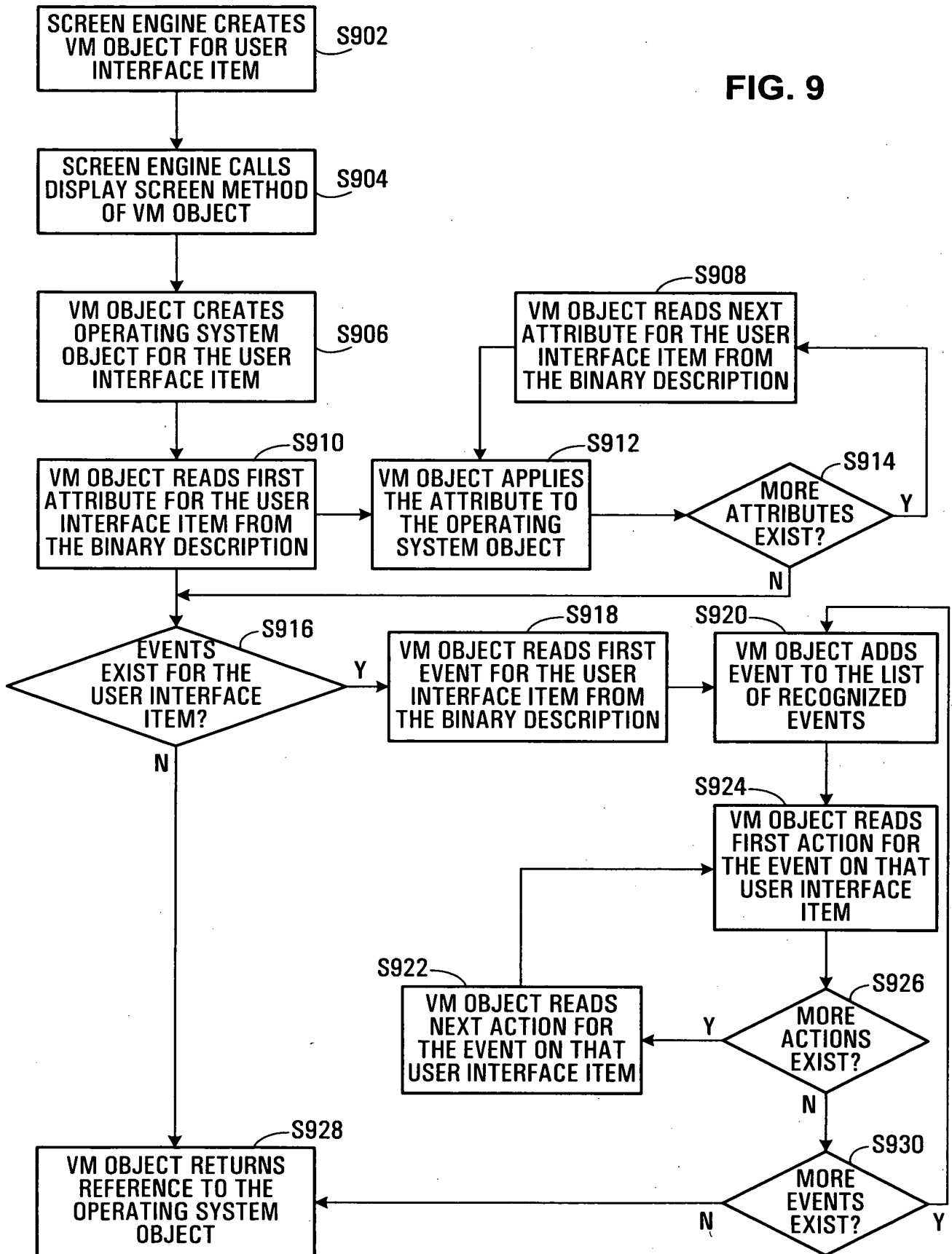


FIG. 8

FIG. 9



10/84

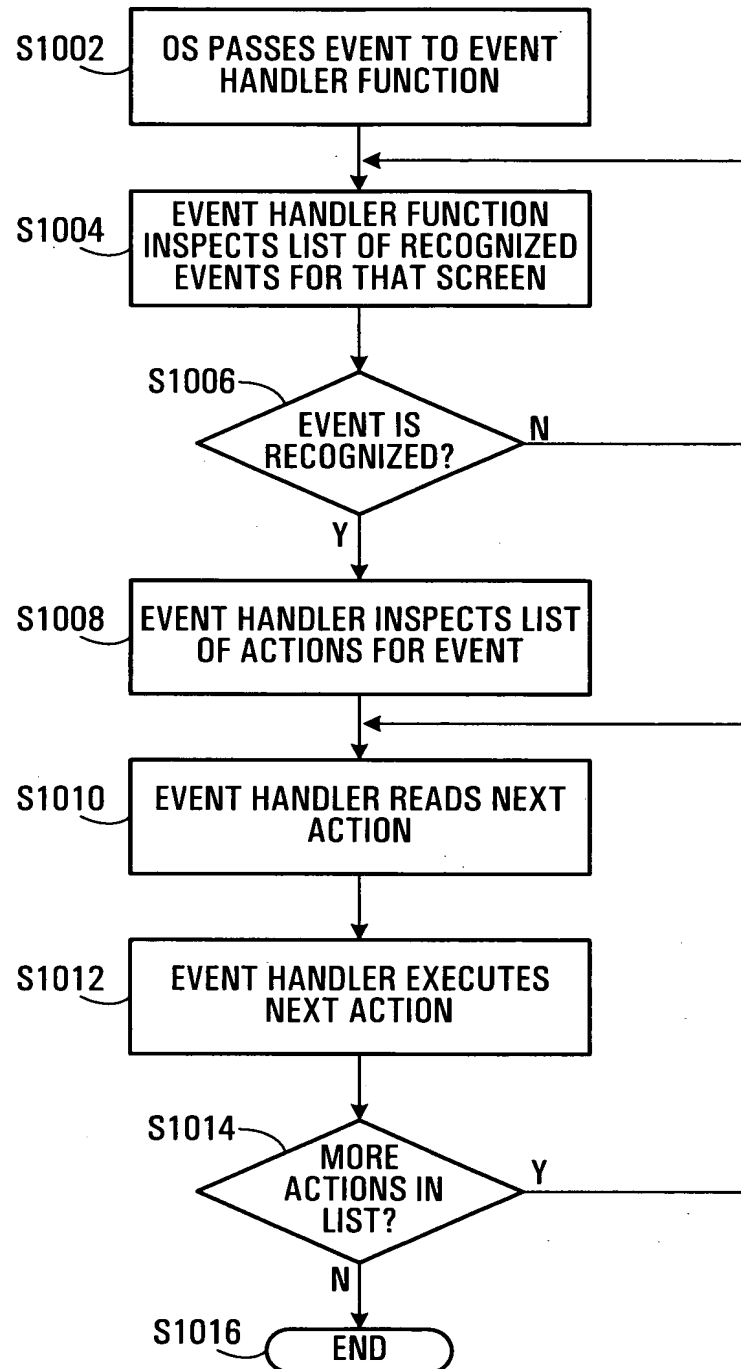


FIG. 10

11/84

```

72 {
  <ARML>
    <HEAD>_</HEAD>
    <SYS>
      <QUERY>
        <PLATFORMS>
          <PLATFORM>WinCE</PLATFORM>
        </PLATFORMS>
      </REG>
    </SYS>
  </ARML>

```

```

74 {
  <ARML>
    <HEAD>_</HEAD>
    <SYS>
      <QUERYRESP>
        <APP>Order Entry</APP>
        <APP>Helpdesk</APP>
        <APP>Engineer Dispatch</APP>
      </QUERYRESP>
    </SYS>
  </ARML>

```

```

76 {
  <ARML>
    <HEAD>_</HEAD>
    <SYS>
      <REG TYPE="ADD">
        <CLIENTID>SUNTRESS</CLIENTID>
        <MOBILEID>867452</MOBILEID>
        <NEWMOBILEID>268625</NEWMOBILEID>
        <PLATFORMS>
          <PLATFORM>WinCE</PLATFORM>
        </PLATFORMS>
      </REG>
    </SYS>
  </ARML>

```

```

76 {
  <ARML>
    <HEAD>_</HEAD>
    <SYS>
      <REGCONFIRM TYPE="ADD">
        <MOBILEID>268625</MOBILEID>
        <VALUE>CONFIRM</VALUE>
        <INTERFACE>
          <BUTTONS NUM="1">
            <BTN NAME="OK" CAPTION="Send" INDEX="0">
              </BTN>
            </BUTTONS>
          <EDITBOXES NUM="3">
            <E3 NAME="To" INDEX="1"></E3>
            <E3 NAME="Subject" INDEX="2"></E3>
            <E3 NAME="Body" INDEX="3"></E3>
          </EDITBOXES>
        </INTERFACE>
      </REGCONFIRM>
    </SYS>
  </ARML>

```

FIG. 11

12/84

```
<EDITBOXES NUM="3">  
  <E3 NAME'="To" INDEX="1"></E3>
```

```
  <E3 NAME'="Subject" INDEX="2"></E3>
```

```
  <E3 NAME'="Body" INDEX="3"></E3>  
</EDITBOXES>
```

```
<BUTTONS NUM="1">  
  <BTN NAME="OK" CAPTION="Send"  
  INDEX="0"></BTN>  
</BUTTONS>
```

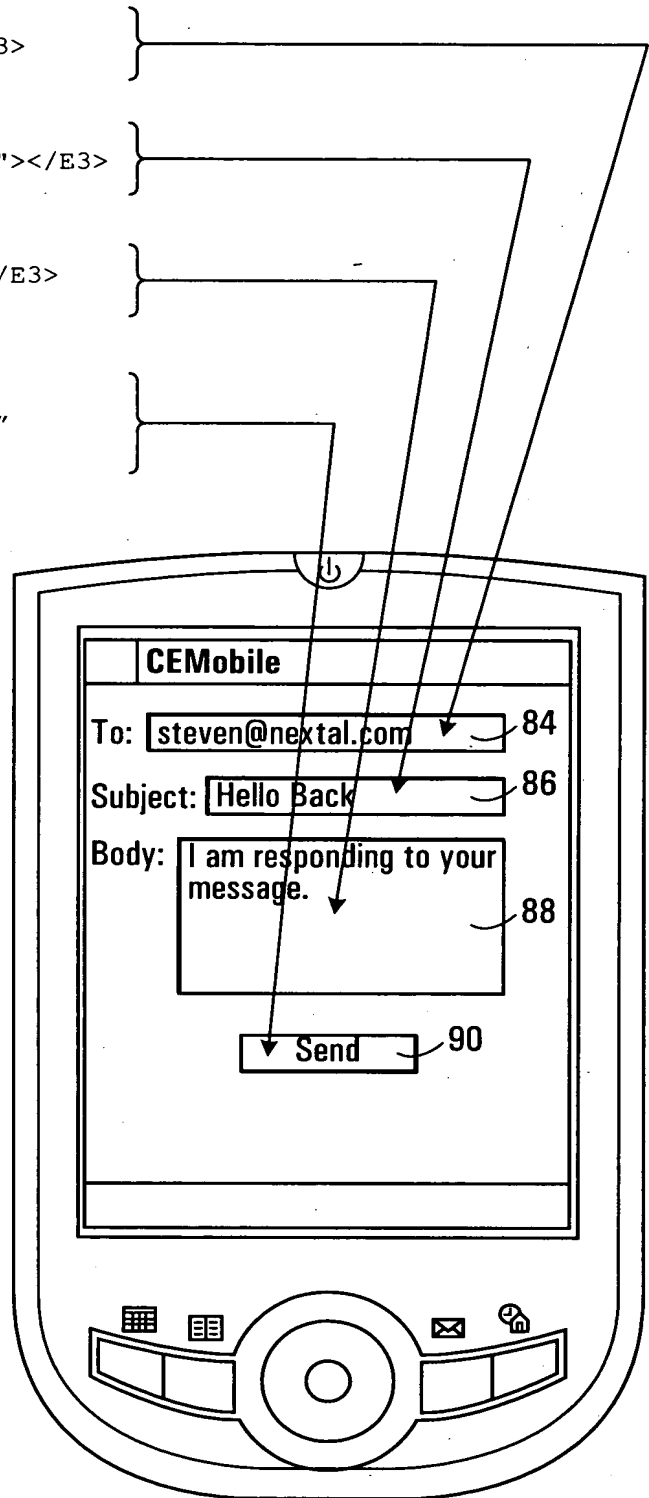




FIG. 12

 1300

```
1 <ARML>  
2   <SYS>  
3     <PING TYPE="SIMPLE"/>  
4   </SYS>  
5 <ARML>
```

FIG. 13

14/84

1400

```
1 <ARML>  
2   <SYS>  
3     <PINGRESP TYPE="SIMPLE"/>  
4   </SYS>  
5 <ARML>
```

FIG. 14

10/537705

15/84

1500

```
1 <ARML>
2   <SYS>
3     <PING TYPE="STATS"/>
4   </SYS>
5 <ARML>
```

FIG. 15

```
1 <ARML>
2   <SYS>
3     <PINGRESP TYPE="STATS"/>Send/Receive: 600
4   Server: http://www.airtrix.com
5   Notifications: Coverage loss, coverage gain, queue backup
6   (5)
7   Smart Client Version: 14.1.0.1
8   Errors:
9     21/10/03 14:22:23 - Could not connect (Send Thread::start)
10    21/10/03 11:38:34 - Illegal argument (AXUIEditBox::Display)
11    21/10/03 10:14:22 - Divide overflow (Field::asString)
12   Queue:
13   <ARML<>SYS<>REG ID="4">mobileid<REG<>/SYS<>/ARML>
14   Queued messages: 1
15   Registered Apps: 4,12
16   Open Screen: LOGIN
17   Make & Model: Blackberry 6710
18   Battery Remaining: 44%
19   Memory Free: 2389 bytes
20   Network: Nextel (IDEN)
21   PIN Number: 4AB328CF2
22   OS: 3.6.1.0
23   Date/Time: 21/10/03 16:11:12
24     </PINGRESP>
25   </SYS>
26 <ARML>
```

FIG. 16

Appendix "A"

Contents

1	Introduction.....	FIG. 17F
1.1	Purpose of document.....	FIG. 17F
1.2	Audience	FIG. 17F
1.3	Definitions & Acronyms.....	FIG. 17F
2	ARML Overview	FIG. 17G
2.1	ARML design considerations	FIG. 17G
2.2	ARML usage.....	FIG. 17H
2.3	The scratchpad area.....	FIG. 17H
2.4	System Variables and Functions.....	FIG. 17I
2.4.1	Variables:.....	FIG. 17I
2.4.2	Functions:.....	FIG. 17I
2.5	Single-Field Lookup	FIG. 17J
3	ARML application definition.....	FIG. 17K
3.1	General.....	FIG. 17K
3.1.1	Description.....	FIG. 17K
3.1.2	Structure.....	FIG. 17K
3.1.3	Tags.....	FIG. 17K
3.2	Table Definitions Section	FIG. 17M
3.2.1	Description.....	FIG. 17M
3.2.2	Structure.....	FIG. 17M
3.2.3	Tags.....	FIG. 17M
3.2.4	Example	FIG. 17N
3.3	Package Definitions Section	FIG. 17P
3.3.1	Description.....	FIG. 17P
3.3.2	Structure.....	FIG. 17P
3.3.3	Tags.....	FIG. 17P
3.3.4	Example	FIG. 17S
3.4	Device Interface Definitions Section.....	FIG. 17T
3.4.1	Description.....	FIG. 17T
3.4.2	Structure.....	FIG. 17T
3.4.3	Tags.....	FIG. 17T
3.4.4	Example	FIG. 17V

FIG. 17A

Appendix "A"

4	Application-defined packages.....	FIG. 17W
4.1	General.....	FIG. 17W
4.1.1	Description.....	FIG. 17W
4.1.2	Structure.....	FIG. 17W
4.1.3	Tags.....	FIG. 17W
4.2	Package information	FIG. 17X
4.2.1	Example	FIG. 17X
5	User interface Definitions.....	FIG. 17Z
5.1	General.....	FIG. 17Z
5.1.1	Description.....	FIG. 17Z
5.1.2	Structure.....	FIG. 17Z
5.1.3	Tags.....	FIG. 17AA
5.2	Queries definition section	FIG. 17CC
5.2.1	Description.....	FIG. 17CC
5.2.2	Structure.....	FIG. 17CC
5.2.3	Tags.....	FIG. 17CC
5.3	Menu definition section	FIG. 17DD
5.3.1	Description.....	FIG. 17DD
5.3.2	Structure.....	FIG. 17DD
5.3.3	Tags.....	FIG. 17DD
5.4	Buttons definition section	FIG. 17FF
5.4.1	Description.....	FIG. 17FF
5.4.2	Structure.....	FIG. 17FF
5.4.3	Tags.....	FIG. 17FF
5.5	Text Items definition section	FIG. 17GG
5.5.1	Description.....	FIG. 17GG
5.5.2	Structure.....	FIG. 17GG
5.5.3	Tags.....	FIG. 17GG
5.6	Edit boxes definition section.....	FIG. 17HH
5.6.1	Description.....	FIG. 17HH
5.6.2	Structure.....	FIG. 17HH
5.6.3	Tags.....	FIG. 17II
5.7	Choice items definition section.....	FIG. 17JJ
5.7.1	Description.....	FIG. 17JJ
5.7.2	Structure.....	FIG. 17JJ
5.7.3	Tags.....	FIG. 17KK
5.8	Checkboxes definition section	FIG. 17LL

FIG. 17B

Appendix "A"

5.8.1	Description.....	FIG. 17LL
5.8.2	Structure.....	FIG. 17LL
5.8.3	Tags.....	FIG. 17MM
5.9	Listboxes definition section	FIG. 17NN
5.9.1	Description.....	FIG. 17NN
5.9.2	Structure.....	FIG. 17NN
5.9.3	Tags.....	FIG. 17NN
5.10	Grids.....	FIG. 17PP
5.10.1	Description.....	FIG. 17PP
5.10.2	Structure.....	FIG. 17PP
5.10.3	Tags.....	FIG. 17QQ
5.10.4	Example	FIG. 17RR
6	The Smart Client event model	FIG. 17TT
6.1	The EVENTS tag	FIG. 17UU
6.2	The EVENT tag	FIG. 17UU
6.2.1	The BUTTONCLICK event	FIG. 17UU
6.2.2	The MENUITEMSELECTED event	FIG. 17UU
6.2.3	The DATA event.....	FIG. 17UU
6.3	The ACTION tag	FIG. 17UU
6.3.1	The OPEN action	FIG. 17VV
6.3.2	The ARML action.....	FIG. 17VV
6.3.3	The SAVE action	FIG. 17VV
6.3.4	The PURGE action	FIG. 17WW
6.3.5	The NOTIFY action.....	FIG. 17WW
6.3.6	The CLOSE action.....	FIG. 17WW
6.3.7	The ALERT action.....	FIG. 17WW
6.3.8	The INTEGRATION action	FIG. 17WW
6.3.9	The CLOSESCREEN action.....	FIG. 17XX
6.3.10	The REFRESH action.....	FIG. 17XX
6.3.11	The SAVEITEM action	FIG. 17XX
6.3.12	The IF Action.....	FIG. 17XX
	Example of airix event model	FIG. 17BBB
7	AVM-server system interactions	FIG. 17DDD
7.1	General.....	FIG. 17DDD
7.1.1	Description.....	FIG. 17DDD
7.1.2	Structure.....	FIG. 17DDD
7.1.3	Tags.....	FIG. 17DDD

FIG. 17C

Appendix "A"

7.2	Device Registration & deregistration package	FIG. 17EEE
7.2.1	Description	FIG. 17EEE
7.2.2	Structure	FIG. 17EEE
7.2.3	Tags	FIG. 17EEE
7.2.4	Example	FIG. 17EEE
7.3	Registration confirmation package	FIG. 17FFF
7.3.1	Description	FIG. 17FFF
7.3.2	Structure	FIG. 17FFF
7.3.3	Tags	FIG. 17FFF
7.3.4	Example	FIG. 17GGG
7.4	Find applications package	FIG. 17HHH
7.4.1	Description	FIG. 17HHH
7.4.2	Structure	FIG. 17HHH
7.4.3	Tags	FIG. 17HHH
7.5	Find applications confirmation package	FIG. 17III
7.5.1	Description	FIG. 17III
7.5.2	Structure	FIG. 17III
7.5.3	Tags	FIG. 17III
7.6	Application Registration & deregistration package	FIG. 17JJJ
7.6.1	Description	FIG. 17JJJ
7.6.2	Structure	FIG. 17JJJ
7.6.3	Tags	FIG. 17JJJ
7.7	Application registration & deregistration confirmation package	FIG. 17KKK
7.7.1	Description	FIG. 17KKK
7.7.2	Structure	FIG. 17KKK
7.7.3	Tags	FIG. 17KKK
7.7.4	Example	FIG. 17LLL
7.8	Setting the active device package	FIG. 17MMM
7.8.1	Description	FIG. 17MMM
7.8.2	Structure	FIG. 17MMM
7.8.3	Tags	FIG. 17MMM
7.8.4	Example	FIG. 17MMM
7.9	Set active device response	FIG. 17MMM
7.9.1	Description	FIG. 17MMM
7.9.2	Structure	FIG. 17MMM
7.9.3	Tags	FIG. 17NNN
7.9.4	Example	FIG. 17NNN

FIG. 17D

Appendix "A"

7.10	Invalid Application package	FIG. 17NNN
7.10.1	Description	FIG. 17NNN
7.10.2	Structure	FIG. 17NNN
7.10.3	Tags	FIG. 17000
7.10.4	Example	FIG. 17000
8	Application-server system interactions	FIG. 17PPP

FIG. 17E

Appendix "A"

1 Introduction

1.1 Purpose of document

This document describes the structure and syntax of the ARML language.

1.2 Audience

The document is intended to be read by AIRIX developers and users of ARML.

1.3 Definitions & Acronyms

ARML	AIRIX Markup Language
XML	Extensible Markup Language

FIG. 17F

Appendix "A"

2 ARML Overview

ARML is an XML markup language used by the AIRIX platform. It performs three tasks;

- Data is passed back and forth between the mobile server, AIRIX platform and enterprise application using ARML.
- The AIRIX Smart Client uses ARML to define the user interface for an AIRIX-enabled application on the mobile device
- The AIRIX server uses ARML to define that data that it stores for the application in its database.

2.1 ARML design considerations

ARML has been designed with the following goals in mind;

- Transactions and screen definitions should be as independent as possible
- AIRIX should be unaware of internals of the enterprise application
- Strict conformance to the XML specification will be enforced
- Operation should be transparent to the end user
- ARML packages should be readable as is
- The minimum number of characters needed should be used

FIG. 17G

Appendix "A"

2.2 ARML usage

The diagram below illustrates how ARML is used.

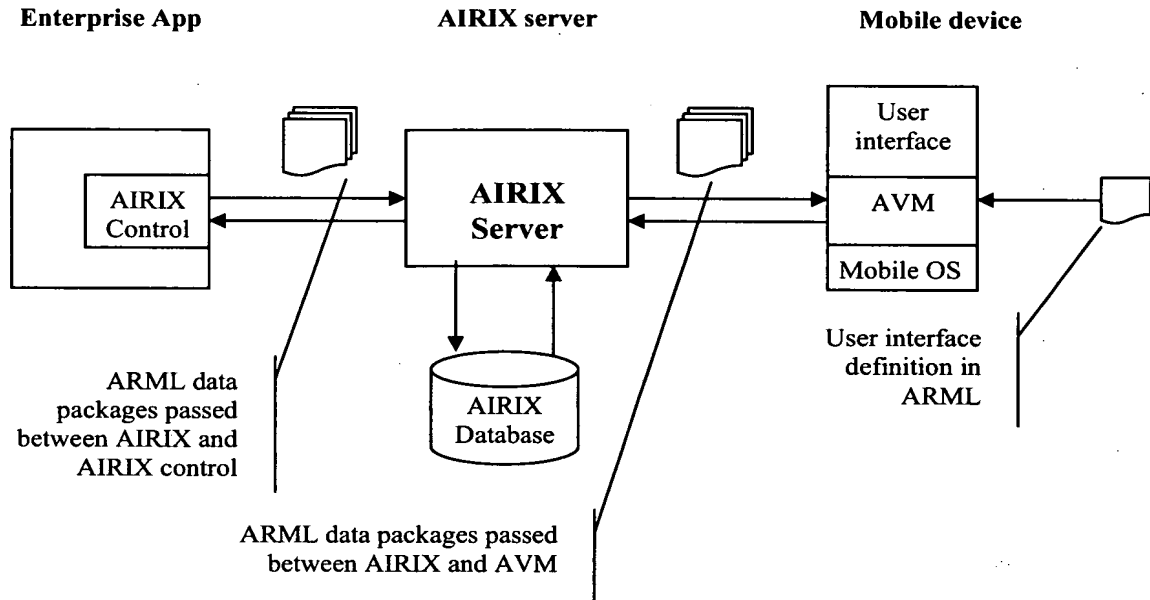


Figure 1 -The ARML environment

The key to ARML usage is the application definition file held on the AIRIX server. This file defines the AIRIX tables for the application, the allowed message set and the user interface definitions for the application on a given device.

2.3 The scratchpad area

The scratchpad is used as a temporary storage area where a global value or a value associated to a screen can be saved for future use. The syntax for a scratchpad value is as follows:

screen scratchpad value: [SP.screen.savename]

FIG. 17H

Appendix "A"

global scratchpad value: [SP.*.savename]

The syntax for retrieving a global scratchpad value can also be used to retrieve screen scratchpad values.

2.4 System Variables and Functions

There are several variables that are available that will retrieve application and system values to be used throughout the application. The syntax for these variables are as follows:

2.4.1 Variables:

[DATE] – returns the current system date, formatted as dd mmm yy

[TIME] – returns the current system time, formatted as hh:mm:ss am/pm.

[SYS.VAR.DATE] - returns the current system date, formatted as dd mmm yy

[SYS.VAR.MOBILEID] - retrieves the device's Mobile ID

[SYS.VAR.APPNAME] - retrieves the name of the application.

[SYS.VAR.APPVERSION] - retrieves the version number of the application.

[SYS.VAR.SCVERSION] - retrieves the version number of the Smart Client.

[SYS.VAR.ARMLMAJOR] - retrieves the ARML major version of the application.

[SYS.VAR.ARMLMINOR] - retrieves the ARML minor version of the application.

2.4.2 Functions:

[SYS.FUNC.DATEADD([SYS.VAR.DATE],+-x)] - The Date Arithmetic tag is used to add or subtract days from the current date. In the tag, x represents the number of days added or subtracted. Developers can also choose to substitute a hard-coded date value in the Date Arithmetic tag, in the place of the [SYS.VAR.DATE] tag.

[SYS.FUNC.DATETOSTR([SYS.VAR.DATE],d mmm yyyy h:nn:ss tz)] - The Date To String tag is used to convert date data to a string value.

[SYS.FUNC.STRTODATE([SYS.VAR.DATE],d mmm yyyy h:nn:ss tz)] - The String to Date tag is used to convert string data to a date value, in the RFC 1123 format.

FIG. 17I

Appendix "A"

2.5 Single-Field Lookup

The single-field lookup will run a simple SELECT query with one where clause to retrieve specific data. The syntax is as follows:

```
[DB.DOLOOKUP(table, field, wherefield, wherevalue)]
```

FIG. 17J

Appendix “A”

3 ARML application definition

3.1 General

3.1.1 Description

The application definition section defines the AIRIX tables and ARML data packages that are used for transactions involved with a specific application.

3.1.2 Structure

The ARML application definition has the following structure;

```
<ARML>
  <AXSCHDEF>
    <EVENTS>
      <EVENT>
        (action definitions)
      </EVENT>
    </EVENTS>
    <AXTDEFS>
      (table definitions)
    </AXTDEFS>
    <DPACKETS>
      (data package definitions)
    </DPACKETS>
    <DEVICES>
      (device interface definitions)
    </DEVICES>
  </AXSCHDEF>
</ARML>
```

3.1.3 Tags

3.1.3.1 The <AXSCHDEF> tag

These tags (<AXSCHDEF>...</AXSCHDEF>) mark the start and end of the application definition. THE AXSCHDEF tag has two attributes;

Attribute	Optional?	Description
APPNAME	No	The name of the application
VERSION	No	Which version of the application the file describes
DESC	No	A text description of the application for display purposes
ARMLMAJOR	No	The major version of the ARML language this application definition was created with.

FIG. 17K

Appendix "A"

ARMLMINOR	No	The minor version of the ARML language this application definition was created with.
-----------	----	--------------------------------------------------------------------------------------

3.1.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

3.1.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

3.1.3.4 The <AXTDEFS> tag

The <AXTDEFS>...</AXTDEFS> pair marks the start and end of the table definitions section. It has no attributes.

3.1.3.5 The <DPACKETS> tag

The <DPACKETS>...</DPACKETS> pair marks the start and end of the data package definitions section. It has no attributes.

3.1.3.6 The <DEVICES> tag

The <DEVICES>...</DEVICES> pair marks the start and end of the device interface definitions section. It has no attributes.

FIG. 17L

Appendix “A”

3.2 Table Definitions Section

3.2.1 Description

The table definitions section defines the tables on the mobile device for the application

3.2.2 Structure

The table definitions section has the following structure;

```
{wrapper tags}
<TDEF>
    <FIELDS>
        <FLD>...</FLD>
    </FIELDS>
</TDEF>
(etc.)
{wrapper tags}
```

3.2.3 Tags

3.2.3.1 The <TDEF> tag

Each table definition is enclosed within the <TDEF>...</TDEF> pair. The TDEF tag has the following attributes;

Attribute	Optional?	Description
NAME	No	The number of table definitions in the section
PK	No	Which of the table fields is the primary key for the table
DELINDEX	No	The index of this table with respect to all the tables for specifying the delete order. This value is 1 based.

3.2.3.2 The <FIELDS> tag

The <FIELDS>...</FIELDS> tag pair marks where the fields in a given table are defined. The FIELDS tag has a no attributes.

3.2.3.3 The <FLD> tag

The <FLD>...</FLD> tag pair defines a single field in a table. Enclosed between the tags is the field name. The <FLD> tag has the following structure;

Attribute	Optional?	Description
TYPE	No	The data type contained in the field. Permitted values are:

FIG. 17M

Appendix "A"

		INT – integer value STRING – a fixed-length string of n characters (see SIZE field) MEMO – a string of max 65535 characters AUTOINC – an integer value, automatically incremented by the database. This field will be read-only to the applications. DATETIME – a datetime value
SIZE	No	If the TYPE is set to STRING, this field specifies the number of characters in the field
INDEXED	No	Specifies if the field needs to be indexed in the AIRIX database
REFERENCEFIELD	Yes	If this attribute is present, it defines that this field is a foreign key. The foreign table/field is given in the format "table(field)"
ALLOWNULL	No	Specifies if the field is allowed to have a null value

3.2.4 Example

An email application would use 2 tables for storing sent emails.

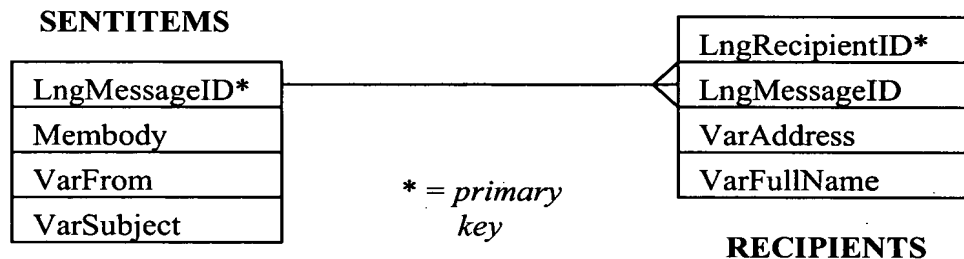


Figure 2 - sample email schema

This translates into the following ARML fragment;

```

<TDEF NAME="SENTITEMS" UPDATETYPE=NEW PK=LNGMESSAGEID DELINDEX=2>
  <FIELDS>
    <FLD TYPE="INT" SIZE="0" INDEXED="NO" REFERENCEFIELD=""
      ALLOWNULL="NO">LNGMESSAGEID</FLD>
    <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
      ALLOWNULL="YES">VARFROM</FLD>
    <FLD TYPE="MEMO" SIZE="0" INDEXED="NO" REFERENCEFIELD=""
      ALLOWNULL="YES">MEMBODY</FLD>
    <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
      ALLOWNULL="YES">VARSUBJECT</FLD>
  </FIELDS>
</TDEF>
<TDEF NAME="RECIPIENTS" UPDATETYPE=NEW PK=LNGRECIPIENTID DELINDEX=1>
  <FIELDS>
    <FLD TYPE="INT" SIZE="AUTOINC" INDEXED="NO" REFERENCEFIELD=""
      ALLOWNULL="NO">LNGMESSAGEID</FLD>
  </FIELDS>
</TDEF>

```

FIG. 17N

Appendix "A"

```
<FLD TYPE="INT" SIZE="0" INDEXED="YES"
    REFERENCEFIELD="SENTITEMS (MESSAGEID) "
    ALLOWNULL="NO">LNGMESSAGEID</FLD>
<FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
    ALLOWNULL="YES">VARFULLNAME</FLD>
<FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
    ALLOWNULL="YES">VARADDRESS</FLD>
</FIELDS>
</TDEF>
```

Figure 3 - a sample table definition section

Appendix “A”

3.3 Package Definitions Section

3.3.1 Description

The package definitions section defines the structure of the application packages and the data that they carry.

3.3.2 Structure

The package definitions section has the following structure;

```
{wrapper tags}
<AXDATAPACKET>
  <TABLEUPDATES>
    <TUPDATE>
      <PKGFIELDS>
        <PKGFLD>...</PKGFLD>
        <PKGFLD>...</PKGFLD>
      </PKGFIELDS>
    </TUPDATE>
  </TABLEUPDATES>
  <TABLEUPDATES>
    <TUPDATE>
      <PKGFIELDS>
        <PKGFLD>...</PKGFLD>
        <PKGFLD>...</PKGFLD>
        (etc.)
      </PKGFIELDS>
    </TUPDATE>
  </TABLEUPDATES>
  (etc.)
</AXDATAPACKET>
{wrapper tags}
```

3.3.3 Tags

3.3.3.1 The <AXDATAPACKET> tag

The <AXDATAPACKET>...</AXDATAPACKET> pair delimits a package definition. The tag has the following attributes;

Attribute	Optional?	Description
BODY	No	This field gives the name by which the data package is known
UPDATELOCALDATA	No	Specifies whether the package is to update the local database.
SENDTOAPP	No	Specifies whether the package is sent to the application server

FIG. 17P

Appendix "A"

3.3.3.2 The <TABLEUPDATES> tag

The <TABLEUPDATES>...</TABLEUPDATES> pair marks the start and end of the table definitions section. It has no attributes.

3.3.3.3 The <TUPDATE> tag

Each table update is enclosed within the <TUPDATE>...</TUPDATE> pair. The TUPDATE tag has the following attributes;

Attribute	Optional?	Description
TABLE	No	The table in the database that is updated
UPDATETYPE	No	The type of update that is being made to the database. Possible values are; ADD – adds a new record into the table DELETE – removes a record into the table UPDATE – modifies a record in the table
WHEREFIELD	Yes	For a conditional update of a table, specifies the field and table to match on. This is in the format "table(field)".
WHEREPARAM	Yes	Text string specifying the value. This tag has no meaning and will be skipped unless the WHEREFIELD attribute has been specified.
SECTION	No	An identifier for the section in the data package
MULTIROW	No	Boolean field specifying whether multiple rows can be updated by the tag
MULTIROWIDENT	Yes	If the MULTIROW attribute is set to 'YES', this field is required and specifies the

3.3.3.4 The <PKGFIELDS> tag

The <PKGFIELDS>...</PKGFIELDS> tag pair marks where the fields in a given data package are defined. The PKGFIELDS tag has no attributes.

3.3.3.5 <The PKGFLD> tag

The <PKGFLD>...</PKGFLD> tag pair defines a single parameter in a given data package. Enclosed between the <PKGFLD>...</PKGFLD> tags is the field name. The <PKGFLD> tag has the following attributes;

Attribute	Optional?	Description
NAME	No	This is the field in the AIRIX database that maps to the user

FIG. 17Q

Appendix "A"

		interface field
PARAMTYPE	No	This defines the type of parameter. It can take two values; PROP – this means that the parameter appears as part of the tag definition VALUE – this means that the parameter is contained between the two tags. Only one parameter in a given data package can be of this type

FIG. 17R

Appendix "A"

3.3.4 Example

Using the table definitions example in section 3.2.4, when the user sends an email, a data package to transport the data would update the 'SENTITEMS' table and the 'RECIPIENTS' table. The following ARML fragment defines such a data package;

```
<AXDATAPACKET BODY="ME" SENDTOMOBILE="NO" SENDTOAPP="YES">
  <TABLEUPDATES>
    <TUPDATE TABLE="SENTITEMS" UPDATETYPE="ADD" WHEREFIELD=""
      WHEREPARAM=""
        WHERETYPE="PROP" SECTION="MAIL" MULTIROW="NO" MULTIROWIDENT="">
      <FIELDS>
        <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
        <PKGFLD NAME="VARFROM" PARAMTYPE="PROP">FROM</PKGFLD>
        <PKGFLD NAME="VARSUBJECT" PARAMTYPE="PROP">SUBJECT</PKGFLD>
        <PKGFLD NAME="MEMBODY" PARAMTYPE="VALUE">DATA</PKGFLD>
      </FIELDS>
    </TUPDATE>
    <TUPDATE TABLE="RECIPIENTS" UPDATETYPE="ADD" WHEREFIELD=""
      WHEREPARAM=""
        WHERETYPE="PROP" SECTION="RECIPS" MULTIROW="YES"
        MULTIROWIDENT="RCP">
      <FIELDS>
        <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
        <PKGFLD NAME="VARFULLNAME" PARAMTYPE="PROP">TO</PKGFLD>
        <PKGFLD NAME="VARADDRESS" PARAMTYPE="PROP">ADDRESS</PKGFLD>
      </FIELDS>
    </TUPDATE>
  </TABLEUPDATES>
</AXDATAPACKET>
```

Figure 4 - a sample package definition

FIG. 17S

Appendix “A”

3.4 Device Interface Definitions Section

3.4.1 Description

The display definitions section contains the user interface definitions for the various mobile devices that an application supports.

3.4.2 Structure

The device display definitions section has the following structure;

```
{wrapper tags}
<DEV>
    <SCREENS>
        <SCREEN>
            {screen definitions}
        </SCREEN>
    </SCREENS>
</DEV>
{other devices}
{wrapper tags}
```

3.4.3 Tags

3.4.3.1 The <DEV> tag

The <DEV>...</DEV> pair delimits an interface definition for a specific device. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	The type of device. Allowed values are: RIM – a Research in Motion Blackberry pager WAP – a WAP phone CE – Pocket PC

3.4.3.2 The <SCREENS> tag

The <SCREENS>...</SCREENS> pair delimits the screens definition for a specific device. The tag has one attribute;

Attribute	Optional?	Description
STSCRN	No	The first screen that is displayed when the application starts

FIG. 17T

Appendix “A”

3.4.3.3 The <SCREEN> tag

The <SCREEN>...</SCREEN> pair, and its contents are described in section 5.1.3.1

FIG. 17U

Appendix "A"

3.4.4 Example

The following example shows the screen definitions section for an application that allows a user to view their inbox and the mails in it.

```
{wrapper tags}
<DEV TYPE="RIM">
  <SCREENS>
    <SCREEN NAME="INBOX ">
      {screen definition}
    </SCREEN>
    <SCREEN NAME="VIEWNEWMAIL">
      {screen definition}
    </SCREEN>
  </SCREENS>
</DEV>
<DEV TYPE="PALM">
  <SCREENS>
    <SCREEN NAME="INBOX">
      {screen definition}
    </SCREEN>
    <SCREEN NAME="VIEWNEWMAIL">
      {screen definition}
    </SCREEN>
  </SCREENS>
</DEV>
{wrapper tags}
```

FIG. 17V

Appendix "A"

4 Application-defined packages

This section describes the format of application defined packages.

4.1 General

This section describes the general structure of an application-specific data package. As described in section , ;

4.1.1 Description

System level packages are sent between AIRIX and the application server, and between AIRIX and the AVM

4.1.2 Structure

An application defined package has the following structure;

```
<ARML>
  <HEAD>
    (header information)
  </HEAD>
  <PKG>
    (package information)
  </PKG>
</ARML>
```

4.1.3 Tags

4.1.3.1 The <HEAD> tag

The <HEAD> tag is as described in section 7.1.3.1

4.1.3.2 The <PKG> tag

The <PKG>...</PKG> tags delimit the package data. The PKG tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	A text string identifying the type of package being sent

FIG. 17W

Appendix "A"

4.2 Package information

The format and rules for application-defined data packages depend on the package definitions for that application.

4.2.1 Example

A sample data package following the rules in section 3.3.4 would have a body section like this;

```
{wrapper tags}
<PKG TYPE="ME">
  <MAIL MSGID="1" FROM="Tim Neil" FROMADDRESS="timn@nextair.com"
    SUBJECT="Hello Back">
    <DATA>I am responding to your message</DATA>
  </MAIL>
  <RECIPS>
    <RCP MSGID="1" TO="Jeff Jones"
      ADDRESS="jeff@nextair.com"></RCP>
    <RCP MSGID="1" TO="Scott Neil"
      ADDRESS="scottn@nextair.com"></RCP>
    <RCP MSGID="1" TO="Steve Hulaj"
      ADDRESS="steveh@nextair.com"></RCP>
  </RECIPS>
</PKG>
{wrapper tags}
```

Figure 5 - a sample package

We will use this sample package to illustrate how packages are derived from the package definition file. The first tag in the package is the BODY tag. This tag defines which type of package it is;

Package Definition

```
<AXDATAPACKET BODY="ME" UPDATELOCALDATA="NO"
```

Package

```
<BODY TYPE="ME">
```

The package has two sections, which correspond to the two table update sections in the package definition;

Appendix "A"

Package Definition

```

<TUPDATE TABLE="SENTITEMS" UPDATETYPE="ADD" WHEREFIELD="" WHEREPARAM=""
  WHEREATYPE="PROP" SECTION="MAIL" MULTIROW="NO" MULTIROWIDENT="">

<TUPDATE TABLE="RECIPIENTS" UPDATETYPE="ADD" WHEREFIELD="" WHEREPARAM=""
  WHEREATYPE="PROP" SECTION="RECIPS" MULTIROW="YES"
  MULTIROWIDENT="RCP">
  Package
    <MAIL MSGID="1" FROM="Tim Neil"
    <RECIPS>
      <RCP>
        <RCP>
          <RCP>
        </RCP>
      </RECIPS>

```

The 'MAIL' section updates the 'SENTITEMS' table in the database. It does not update multiple rows. The 'RECIPS' section updates the 'RECIPIENTS' table in the database; it does update multiple rows, and each row is contained within a pair of <RCP> tags.

Each of the MAIL and RCP tags have fields which are used to update the field in the database tables;

Package Definition

```

<FIELDS>
  <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
  <PKGFLD NAME="VARFULLNAME" PARAMTYPE="PROP">TO</PKGFLD>
  <PKGFLD NAME="VARADDRESS" PARAMTYPE="PROP">ADDRESS</PKGFLD>
</FIELDS>

Package
  <RCP MSGID="1" TO="Jeff Jones" ADDRESS="jeff@nextair.com"></RCP>

```

FIG. 17Y

Appendix "A"

5 User interface Definitions

5.1 General

5.1.1 Description

A screen definition file defines a single screen for a specific device.

5.1.2 Structure

A screen definition file has the following structure;

```
<ARML>
  <SCREEN>
    <EVENTS>
      <EVENT>
        <ACTION>...</ACTION>
      </EVENT>
    </EVENTS>
    <QUERIES>
      (menu definition)
    </QUERIES>
    <MENUS>
      (menu definition)
    </MENUS>
    <BUTTONS>
      (button definitions)
    </BUTTONS>
    <TEXTITEMS>
      (textitem definitions)
    </TEXTITEMS>
    <EDITBOXES>
      (edit box definitions)
    </EDITBOXES>
    <CHOICEITEMS>
      (choice item definitions)
    </CHOICEITEMS>
    <IMAGES>
      (image definitions)
    </IMAGES>
    <LISTBOXES>
      (list box definitions)
    </LISTBOXES>
    <CHECKBOXES>
      (check box definitions)
    </CHECKBOXES>
    <GRIDS>
      (check grid definition)
```

FIG. 17Z

Appendix “A”

```

        </GRIDS>
    </SCREEN>
</ARML>

```

5.1.3 Tags

5.1.3.1 The SCREEN tag

The <SCREEN>...</SCREEN> pair marks the start and end of the screen definitions section. It has attribute –

Attribute	Optional?	Description
NAME	No	An identifier for the screen. This is used to qualify variables and navigate between screens
TITLE	No	The title that appears for the screen.
BACKGROUND	Yes	If used, an image that appears behind the interface elements
ORDERED	Yes, only applicable on WAP	If yes, WML is created with ORDERED property set to true, if NO, WML is created with ORDERED property set to false. Only applicable on WAP. See WML standard for definition of ORDERED.

5.1.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.1.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.1.3.4 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.1.3.5 The QUERIES tag

The <QUERIES>...</QUERIES> pair marks the start and end of the queries definitions section. It has no attributes.

FIG. 17AA

Appendix "A"

5.1.3.6 The **MENUS** tag

The `<MENUS>...</MENUS>` pair marks the start and end of the menu definition section. It has no attributes.

5.1.3.7 The **BUTTONS** tag

The `<BUTTONS>...</BUTTONS>` pair marks the start and end of the button definitions section. It has no attributes.

5.1.3.8 The **TEXTITEMS** tag

The `<TEXTITEMS>...</TEXTITEMS>` pair marks the start and end of the text items section. It has no attributes.

5.1.3.9 The **EDITBOXES** tag

The `<EDITBOXES>...</EDITBOXES>` pair marks the start and end of the editboxes section. It has no attributes.

5.1.3.10 The **CHOICEITEMS** tag

The `<CHOICEITEMS>...</CHOICEITEMS>` pair marks the start and end of the choiceitems section. It has no attributes.

5.1.3.11 The **IMAGES** tag

The `<IMAGES>...</IMAGES>` pair marks the start and end of the images section. It has no attributes.

5.1.3.12 The **CHECKBOXES** tag

The `<CHECKBOXES>...</CHECKBOXES>` pair marks the start and end of the checkboxes section. It has no attributes.

5.1.3.13 The **LISTBOXES** tag

The `<LISTBOXES>...</LISTBOXES>` pair marks the start and end of the listboxes section. It has no attributes.

5.1.3.14 The **GRIDS** tag

The `<GRIDS>...</GRIDS>` pair marks the start and end of the grids section. It has no attributes.

FIG. 17BB

Appendix “A”

5.2 Queries definition section

5.2.1 Description

The queries definition section describes any queries that need to be run to populate a screen.

5.2.2 Structure

The queries definition section has the following structure;

```
{wrapper tags}
<QUERIES>
  <QUERY>
    <W>...</W>
  </QUERY>
</QUERIES>
{wrapper tags}
```

5.2.3 Tags

5.2.3.1 The <QUERIES> tag

The <QUERIES> ... </QUERIES> pair marks the start and end of query definition section. It has no attributes.

5.2.3.2 The <QUERY> tag

The <QUERY>...</QUERY> pair marks the start and end of a given query. It has the following attributes;

Attribute	Optional?	Description
NAME	No	Name of the query.
TABLE	No	The table in the database that is updated
ORDERBY	Yes	Specifies the name of the field in the table that the results are to be ordered on.
ORDERDIR	Yes	ASC or DESC, sort ascending or descending respectively. If ORDERBY is present and ORDERDIR is not, ASC is assumed.

FIG. 17CC

Appendix "A"

5.2.3.3 The <W> tag

The <W>...</W> pair marks the start and end of a given where clause. The value of the parameter is contained within the <W>...</W> tags. This value can be a specific value or a reference to a user interface field in the format "[SP.screen.savename] or [QU.query.field]". It has the following attributes;

Attribute	Optional?	Description
F	No	Specifies the field to match on.
E	No	Specifies the expression type. Available expression types include: EQ NE LT GT BW (applicable only to fields of type STRING)

5.3 Menu definition section

5.3.1 Description

The menu definition section describes the menu for a given screen.

5.3.2 Structure

The menu definition section has the following structure;

```
{wrapper tags}

<MENUS>
  <MENU>
    <MENUITEM>
      <EVENTS>
        <EVENT>
          <ACTION>...</ACTION>
        </EVENT>
      </EVENTS>
    </MENUITEM>
  </MENU>
</MENUS>
{wrapper tags}
```

5.3.3 Tags

FIG. 17DD

Appendix “A”

5.3.3.1 The <MENUS> tag

The <MENUS> ... </MENUS> pair marks the start and end of menu definition section. It has no attributes.

5.3.3.2 The <MENU> tag

The <MENU> ... </MENU> pair marks the start and end of a menu definition. It has the following attributes.

Attribute	Optional?	Description
NAME	No	An internal identifier for the menu
CAPTION	No	The text that appears for this item in the menu

5.3.3.3 The <MENUITEM> tag

The <MENUITEM>...</MENUITEM> pair marks the start and end of a menuitem definition. It has the following tags;

Attribute	Optional?	Description
NAME	No	An internal identifier for the menu
CAPTION	No	The text that appears for this item in the menu
INDEX	Yes	The index of this menu item with respect to all of the menu items on this menu.
READONLY	Yes	If True, the menu item is inactive. False is the default.

5.3.3.4 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.3.3.5 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.3.3.6 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

FIG. 17EE

Appendix "A"

5.4 Buttons definition section

5.4.1 Description

The buttons definition section describes the buttons that appear on a given screen.

5.4.2 Structure

The buttons definition section has the following structure;

```
{wrapper tags}
<BTN>
    <EVENTS>
        <EVENT>
            <ACTION>...</ACTION>
        </EVENT>
    </EVENTS>
</BTN>
{wrapper tags}
```

5.4.3 Tags

5.4.3.1 The BTN tag

The <BTN>...</BTN> pair marks the start and end of a button definition. It has one attribute –

Attribute	Optional?	Description
NAME	No	An identifier for the button.
INDEX	No	The order in which the button appears
CAPTION	No	The caption that appears on a given button
X	Yes	The X-coordinate of the button on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the button on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	This is the Height of the button. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	This is the Width of the Button. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
READONLY	Yes	If True, the button is not enabled. False is the default.

FIG. 17FF

Appendix "A"

5.4.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.4.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.4.3.4 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.5 Text items definition section

5.5.1 Description

The text items definition

5.5.2 Structure

The text items section has the following structure;

```
{wrapper tags}
<TI>
    <EVENTS>
        <EVENT>
            <ACTION>...</ACTION>
        </EVENT>
    </EVENTS>
</TI>
{wrapper tags}
```

5.5.3 Tags

5.5.3.1 The TI tag

The <TI>...</TI> pair marks the start and end of the screen definitions section. It has attribute –

Attribute	Optional?	Description
INDEX	No	The order in which the text item appears

FIG. 17GG

Appendix "A"

NAME	No	An Identifier for the Text Item
CAPTION	No	Text to appear on the Text Item
X	Yes	The X-coordinate of the text item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the text item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	This is the Height of the Text Item. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	This is the Width of the Text Item. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser

5.5.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.5.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.5.3.4 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.6 Edit boxes definition section

5.6.1 Description

The edit boxes definition section describes what edit boxes exist for the screen.

5.6.2 Structure

The edit boxes section has the following structure;

FIG. 17HH

Appendix "A"

```

(wrapper tags)
<EB>
    <EVENTS>
        <EVENT>
            <ACTION>...</ACTION>
        </EVENT>
    </EVENTS>
</EB>
(wrapper tags)

```

5.6.3 Tags

5.6.3.1 The EB tag

The <EB>...</EB> pair marks an edit box definition. It has the following attributes –

Attribute	Optional?	Description
NAME	No	An identifier for the edit box.
TEXT	No	The text to display in the edit box before any entry has been made. Only used if the DATASRC attribute is invalid or omitted. Can be a scratchpad or query value of the form [SP.screen.savename] or [QU.query.field].
INDEX	No	The order in which the edit box appears
CAPTION	No	The caption for on a given edit box.
MULTILINE	No	Boolean field that indicates whether the edit box is a multiline field.
SAVE	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
SAVENAME	Yes	If present, the name to save the field under in the scratchpad. This attribute has no meaning unless the SAVE attribute is set to 'Yes'
X	Yes	The X-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	The Height of the Edit Box. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	The Width of the Edit Box. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
FT	Yes	Specifies the type of value expected (INT, STRING,

FIG. 17II

Appendix “A”

		MEMO,DATETIME) for the VM to validate prior to continuing a Save. If omitted, STRING is the default data type.
DATASRC	Yes	If present, the query and field in the query that populates this edit box. This is given in the format “query.field”.
READONLY	Yes	If “TRUE” the edit box will be read only, otherwise it is editable. “FALSE is the default value.

5.6.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.6.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.6.3.4 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.7 Choice items definition section

5.7.1 Description

The choice item definitions section describes the choice items that exist on a given screen. A choice item is an interface item that requires the user to make a selection from a list of options. It can be represented in different ways on different devices; on a RIM pager, it is a choice box, while on a WinCE device, it is a drop-down list.

5.7.2 Structure

The choice items section has the following structure;

```
{wrapper tags}
<CHOICE>
    <EVENTS>
        <EVENT>
            <ACTION>...</ACTION>
        </EVENT>
    </EVENTS>
    <ITEMS>
        <I>...</I>
```

FIG. 17JJ

Appendix “A”

```

    </ITEMS>
</CHOICE>
{wrapper tags}

```

5.7.3 Tags

5.7.3.1 The <CHOICE> tag

The <CHOICE>...</CHOICE> pair marks the start and end of a choice item definition. It has these attributes –

Attribute	Optional?	Description
NAME	No	An identifier for the choice item.
TEXT	No	The text to display in the choice item before any selection has been made.
INDEX	No	The order in which the choice item appears
CAPTION	No	The caption that appears for a given choice item
SAVE	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit ‘SAVE’ action on a user interface control.
SAVENAME	Yes	If present, the name to save the field under in the scratchpad. This attribute has no meaning unless the SAVE attribute is set to ‘Yes’
X	Yes	The X-coordinate of the choice item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the choice item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the query and field in the query that populates this choice item. This is given in the format “query.field”.
IDDATASRC	Yes	If present, the query and field in the query that populates the IDs for this choice item. This is given in the format “query.field”. The ID values created by the attributes should correspond directly to the choice item values. I.e. they should create a value, id pair.
READONLY	Yes	If “True”, the control cannot be modified. “False” is the default.
SI	Yes	The value to indicate which item of the choice item is to be selected when loaded. This value will be compared with the ID property (hard-coded items) or the IDDATASRC property (database items).

FIG. 17KK

Appendix "A"

5.7.3.2 The <ITEMS> tag

The <ITEMS>...</ITEMS> pair marks the start and end of a list of items to be included in the choice item. If a datasrc is specified, the <ITEMS> section is ignored.

5.7.3.3 The <I> tag

The <I>...</I> pair marks the start and end of an individual item in the choice items list. It has the following attributes:

Attribute	Optional?	Description
ID	Yes	An id used to identify this item in the list.

The value between the pair is the text value that is to be displayed in the choice item.

5.7.3.4 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.7.3.5 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.7.3.6 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.8 Checkboxes definition section

5.8.1 Description

The checkboxes section describes a check box that appears on a given screen.

5.8.2 Structure

The checkboxes section has the following structure;

FIG. 17LL

Appendix “A”

```

(wrapper tags)
  <CHK>
    <EVENTS>
      <EVENT>
        <ACTION>...</ACTION>
      </EVENT>
    </EVENTS>
  </CHK>
(wrapper tags)

```

5.8.3 Tags

5.8.3.1 The CHK tag

The <CHK>...</CHK> pair marks a check box definition

Attribute	Optional?	Description
NAME	No	An identifier for the check box.
INDEX	No	The index of this control with respect to the list of all controls on the screen.
CAPTION	No	The text to be displayed for this check box if the DATASRC is not available or is not specified.
Save	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
SAVENAME	Yes	If present, the name to save the field under in the scratchpad. This attribute has no meaning unless the SAVE attribute is set to 'Yes'
X	Yes	The X-coordinate of the check box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the check box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	The Height of the Checkbox. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	The Width of the Checkbox. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the query and field in the query that populates this check box. This is given in the format "query.field".
VALUE	Yes	If present, specifies the initial state of the check box ('TRUE' = checked, 'FALSE' = Unchecked. If unspecified, FALSE is the default value.
READONLY	Yes	If "TRUE" the check box cannot be modified. "FALSE" is the default value.

FIG. 17MM

Appendix "A"

5.8.3.2 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.8.3.3 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.8.3.4 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.9 Listboxes definition section

5.9.1 Description

The listboxes section describes a list box that appears on a given screen.

5.9.2 Structure

The listboxes section has the following structure;

```
{wrapper tags}
  <LB>
    <EVENTS>
      <EVENT>
        <ACTION> ... </ACTION>
      </EVENT>
    </EVENTS>
    <ITEMS>
      <I> ... </I>
    </ITEMS>
  </LB>
{wrapper tags}
```

5.9.3 Tags

5.9.3.1 The LB tag

The <LB>...</LB> pair marks a list box definition

FIG. 17NN

Appendix "A"

Attribute	Optional?	Description
NAME	No	An identifier for the list box.
INDEX	No	The index of this control with respect to all of the controls on the screen.
CAPTION	No	The text to be displayed as the title of this list box, where applicable.
SAVE	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
SAVENAME	Yes	If present, the name to save the field under in the scratchpad. This attribute has no meaning unless the SAVE attribute is set to 'Yes'
X	Yes	The X-coordinate of the list box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the list box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	The Height of the Listbox. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	The Width of the Listbox. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the query and field in the query that populates this list box. This is given in the format "query.field".
IDDATASRC	Yes	If present, the query and field in the query that populates the list box Ids. This is given in the format "query.field". This value will create a list of ID values that correspond to the list box values in DATASRC. I.e. they should create a value, id pair.
READONLY	Yes	If "TRUE" the list box cannot be modified. "FALSE" is the default.
SI	Yes	The value to indicate which item of the choice item is to be selected when loaded. This value will be compared with the ID property (hard-coded items) or the IDDATASRC property (database items).

5.9.3.2 The <ITEMS> tag

The <ITEMS>...</ITEMS> pair marks the start and end of a list of items to be included in the in the list box. If a datasrc is specified, the <ITEMS> section is ignored.

5.9.3.3 The <I> tag

The <I>...</I> pair marks the start and end of an individual item in the list box items list. It has the following attributes:

FIG. 1700

Appendix “A”

Attribute	Optional?	Description
ID	Yes	An id used to identify this item in the list.

The value between the pair is the text value that is to be displayed in the list box. Can be a scratchpad or query value of the form [SP.screen.savename] or [QU.query.field].

5.9.3.4 The <EVENTS> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.9.3.5 The <EVENT> tag

The <EVENT>...</EVENT> pair marks the start and end of a user-interface level event definition. See section 6 for a detailed discussion of the Smart Client event model.

5.9.3.6 The <ACTION> tag

The <ACTION>...</ACTION> pair marks the start and end of an action definition. See section 6 for a detailed discussion of the Smart Client event model.

5.10 Grids

5.10.1 Description

Grids allow data to be displayed in row-column format. Grids can display data from a data source (query) or they can contain hard coded values. Each column in a grid can be visible or hidden. Hidden values are maintained, but not visible to the user.

5.10.2 Structure

The grids section has the following structure;

```
{wrapper tags}
  <GRID>
    <COLS>
      <COL> ... </COL>
    </COLS>
    <ROWS>
      <R>
        <V> ... </V>
```

FIG. 17PP

Appendix "A"

```

        </R>
    </ROWS>
</GRID>
(wrapper tags)

```

5.10.3 Tags

5.10.3.1 GRID Tag

<GRID>...</GRID> The grid item itself will have the following attributes

Attribute	Optional?	Description
NAME	No	An identifier for the edit box.
INDEX	No	The order in which the edit box appears
X	Yes	The X-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
HT	Yes	The Height of the Edit Box. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
WT	Yes	The Width of the Edit Box. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
GRDSRC	Yes	This is the Query on the screen that will provide the data to the grid. No field name will be specified in this value

5.10.3.2 COLS Tag

<COLS>...</COLS> This tag contains no attributes. But instead contains all the columns that are associated with the grid in the order in which they appear from left to right.

5.10.3.3 COL Tag

<COL>...</COL> This tag will determine the column specification for the grid. The attributes for this item are the following:

Attribute	Optional?	Description
-----------	-----------	-------------

FIG. 17QQ

Appendix "A"

CAPTION	Yes	This is the caption that appears at the top of the grid where applicable.
FIELDNAME	Yes	This field name represents the Field to pull information from out of the GRDSRC of the grid control.
SAVE	No	This true false value will be checked when the SAVE action is called to save values to the scratchpad
SAVENAME	Yes	This is the name in which the data will be saved when the SAVE action is called and the column is marked for Saving
WT	Yes	The Width of the Edit Box. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser

5.10.3.4 ROWS Tag

`<ROWS>...</ROWS>` This will Indicate any hard coded rows that would be created in the design studio. It does not contain any attributes but instead contains all the row definitions.

5.10.3.5 R Tag

`<R>...</R>` This is the row declaration that contains all the values for the row that has been hard coded. It has no attributes itself, but contains the value definitions for the row.

5.10.3.6 V Tag

`<V>...</V>` This definition contains the data that is related to the ROW and to the column.

5.10.4 Example

An example of a grid declaration is as follows:

```
<GRID INDEX="2" NAME="mygrid" X="10" Y="50" HT="100" WT="100" GRDSRC="QUERY1">
<COLS>
<COL CAPTION="Id" FIELDNAME="lngID" SAVE="TRUE" SAVENAME="lngID" WT="20"></COL>
<COL CAPTION="Subject" FIELDNAME="strSubject" SAVE="TRUE" SAVENAME="Sub"
WT="80"></COL>
</COLS>
<ROWS>
<R>
  <V>343432</V>
  <V>This is a subject</V>
</R>
<R>
```

FIG. 17RR

Appendix "A"

<V>5456</V>
<V>This is another subject</V>
</R>
</ROWS>
</GRID>

FIG. 17SS

Appendix "A"

6 The Smart Client event model

The Smart Client has a set of actions that it ties to events. Events can occur at the application level, the screen level or the user interface item level; an application level event is listened for throughout the operation of the application, a screen level event is listened for while the screen is displayed, and so on. If an action for an event is defined at multiple levels, the lowest level has precedence; i.e., user interface actions override screen level actions, which override application level actions. An attempt to list an event multiple times at the same level (application, screen, item) is invalid and will generate an error message.

The following ARML fragment illustrates this schema (tags and attributes not relevant to the event model have been omitted);

```

<AXTSCHDEF>
  <EVENTS>
    <EVENT>
      <ACTION>...</ACTION>
      <ACTION>...</ACTION>
    <EVENTS>
    <EVENT>
      <ACTION>...</ACTION>
    </EVENT>
  </EVENTS>
  <INTERFACE>
    <SCREEN>
      <EVENT>
        <ACTION>...</ACTION>
      </EVENT>
      <EVENT>
        <ACTION>...</ACTION>
      </EVENT>
      <BUTTON>
        <EVENT>
          <ACTION>...</ACTION>
        </EVENT>
        <EVENT>
          <ACTION>...</ACTION>
        </EVENT>
      </BUTTON>
    </SCREEN>
  </INTERFACE>
</AXTSCHDEF>

```

FIG. 17TT

Appendix "A"

6.1 The EVENTS tag

The <EVENTS>...</EVENTS> pair marks the start and end of the events section. It has no attributes.

6.2 The EVENT tag

The <EVENT>...</EVENT> pair marks the start and end of an event definition. It has the following attributes;

Attribute	Optional?	Description
TYPE	No	The type of event that should be performed when the button is pushed. Allowed values are; BUTTONCLICK MENUITEMSELECTED DATA

6.2.1 The BUTTONCLICK event

The button click event occurs when the user selects a button. It has no attributes.

6.2.2 The MENUITEMSELECTED event

The menu items selected event occurs when the user selects a menu item. It has no attributes.

6.2.3 The DATA event

The data event occurs when ARML data is received from the wireless interface. It has the following attributes;

Attribute	Optional?	Description
NAME	No	The identifier of the specific package

6.3 The ACTION tag

The <ACTION>...</ACTION> pair marks the start and end of an event definition. It has one fixed attribute, and a number of attributes that may or may not appear depending on the type of action required. The fixed attribute is;

Attribute	Optional?	Description
TYPE	No	The type of action that should be performed when the button is pushed.

FIG. 17UU

Appendix "A"

		Allowed values are; OPEN ARML SAVE PURGE NOTIFY CLOSE ALERT IF...Then...Else CLOSESCREEN REFRESH SAVEITEM
--	--	--------------------------------------------------------------------------------------------------------------------------------------------

6.3.1 The OPEN action

The open action tells the Smart Client to open a new screen. It adds one extra attribute to the ACTION tag;

Attribute	Optional?	Description
NAME	No	The name of the screen to open
NEWINST	Yes	If true, a new instance of the screen is created. If false, the least recently used instance of the screen is opened and the data is not refreshed. True is the default.

6.3.2 The ARML action

The arml action tells the Smart Client to compose and send an arml package. It does not add any attributes to the ACTION tag, but has the following subtag;

<ARMLTEXT>

Contained between the *<ARMLTEXT>...</ARMLTEXT>* pair is one of the application-defined data packages. Individual data items are marked with the user interface item that their value should be taken from, in the format "[SP.screen.savename]", or [QU.query.field]. If *screen* is not the current screen, then the Smart Client will look for the data in its scratchpad. See section 0 for an example of the ARML action.

6.3.3 The SAVE action

The save action tells the Smart Client to save all fields marked as persistent (i.e., they are defined with SAVE="Yes") to be saved to the scratchpad area. It has no attributes.

FIG. 17VV

Appendix "A"

6.3.4 The PURGE action

The purge action tells the Smart Client to clear all fields that have been saved to the scratchpad. It has no attributes.

6.3.5 The NOTIFY action

The notify action tells the Smart Client to activate the configured notification on a device. For devices where this has no meaning, it will cause a beep to be played. It has no attributes.

6.3.6 The CLOSE action

The close action tells the Smart Client to close the application. It has no attributes.

6.3.7 The ALERT action

The alert action tells the Smart Client to display an alert item (e.g., a message box on Windows, an alert box on the RIM pager, an alert card on WAP). It has the following attributes;

Attribute	Optional?	Description
CAPTION	Yes	The caption to display in the title bar of the message box
TEXT	Yes	The text to display in the message box

6.3.8 The INTEGRATION action

The integration action tells the Smart Client to pass data to an interface exposed on a device. For example a COM interface on Pocket PC. This action will allow the developer to pass a parameter into an exposed method and then also save the result of that method in a global scratchpad value. The contents of the integration action's element are the input values to be passed to the interface. It has the following attributes;

Attribute	Optional?	Description
CLSID	No	This is the class identifier of the component that is to be called.
SAVE	No	This tells the smart client if it should save the result into a global scratchpad value or not.
SAVENAME	Yes	This is the name of the global scratchpad value

Example ARML:

FIG. 17WW

Appendix "A"

```
<ACTION TYPE="INTEGRATION" CLSID="AirixSignature.AirixSignatureCtrl"
SAVENAME="" SAVE="FALSE">[SP.*.SIGNATURE]</ACTION>
```

6.3.9 The CLOSESCREEN action

The close screen action tells the Smart Client to close all open instances of the screen specified by name in the NAME attribute. This action has the following attributes:

Attribute	Optional?	Description
NAME	No	Name of the screen to close.

6.3.10 The REFRESH action

The refresh action tells the Smart Client to re-run any queries and re-initialize all UI elements on the screen with the name specified by the NAME attribute. If there are multiple open instances of the screen, all open instances will be refreshed. The refresh action has the following attributes:

Attribute	Optional?	Description
NAME	No	Name of the screen to refresh.

6.3.11 The SAVEITEM action

The saveitem action tells the Smart Client to create a new scratchpad item or to edit an existing scratchpad item. The value of the scratchpad item is defined within the <ACTION>...</ACTION> tags. The saveitem action has the following attributes:

Attribute	Optional?	Description
SPN	No	Name of the scratchpad item to create or modify.

6.3.12 The IF Action

This action will contain two lists of actions. One a list of actions to perform if the condition evaluates to TRUE (IFLIST), and another list of actions to perform if the condition evaluates to FALSE (ELSEIFLIST).

The structure of the action is as follows:

```
<ACTION TYPE="IF">
  <COND EVAL="parameter" TYPE="condition type" VALUE="literal">
  </COND>
  <IFLIST>
```

FIG. 17XX

Appendix "A"

```

        <ACTION></ACTION>
    </IFLIST>
    <ELSEIFLIST>
        <ACTION></ACTION>
    </ELSEIFLIST>
</ACTION>

```

6.3.12.1 Conditions (COND)

Conditions are used in conjunction with the IF Action. Conditions are specified as follows:

Attribute	Optional?	Description
EVAL	NO	Specifies the parameter to be evaluated. Can be hard coded, scratchpad, or query values. It is the "input" to the function.
TYPE	NO	Specifies the type of the condition. Possible values are: LESSTHAN MORETHAN EQUALS ISNUMERIC ISALPHA ISEMAIL ISFORMAT MAXCHARS MINCHARS
VALUE	Depends on TYPE	The value that EVAL will be evaluated against. Not relevant for all conditions.

The following is a description of each of the supported conditions:

- **EQUALS**, this function will take an input and a value to evaluate the input against. If the two items are determined to be Equal, the condition will return true. If they are not equal, the condition will return false. The value and the input must be of the same data type, otherwise the condition will return false. Memo values will be treated as a string and auto-increment types will be treated as integers. The following criteria will be used to determine equality:
 - Two strings are equal if each of the characters in the strings is identical and the strings have the same number of characters. The string comparison will not be case sensitive.
 - Two integers are equal if their values are mathematically equal.
- **MORETHAN**, this function will take an input and a value to evaluate the input against. If the input is determined to be greater in value than the evaluation

FIG. 17YY

Appendix "A"

value, the condition will return true. If the values are equal, false is returned. If the evaluation value is determined to be greater than the input, the function will return false. The evaluation value and the input must be of the same data type, otherwise an error condition will occur. Memo values will be treated as a string and the auto-increment type will be treated as an integer. The following criteria will be used to determine which value is greater:

- String A is more in value than String B if String A occurs before String B in alphabetical order.
 - Integer A is greater than Integer B if $A > B$, mathematically.
- LESSTHAN, this function will take an input and a value to evaluate the input against. If the input is determined to be lesser in value than the evaluation value, the condition will return true. If the values are equal, false is returned. If the evaluation value is determined to be lesser than the input, the function will return false. The evaluation value and the input must be of the same data type, otherwise an error condition will occur. Memo values will be treated as a string and the auto-increment type will be treated as an integer. The following criteria will be used to determine which value is greater:
 - String A is lesser in value than String B if String A occurs after String B in alphabetical order.
 - Integer A is greater than Integer B if $A < B$, mathematically.
- ISNUMERIC, this function will take an input and evaluate whether or not it is a value number. If the input can be converted successfully to a number, the function will return true. If the input cannot be converted to a number, the function will return false. All input values will be treated as a string data type.
- ISALPHA, this function will take an input and evaluate whether or not it contains only alphabetic characters. Alphabetic characters are defined as all characters from A-Z, a-z, and spaces. All input values will be treated as a string data type.
- ISEMAIL, this function will take an input and evaluate whether or not it contains a string of the form something@something. All input values will be treated as a string data type.
- ISFORMAT, this function will take an input and a value to evaluate the input against. If the input is determined to be formatted as the evaluation value, the condition will return true. If the evaluation value is determined to be formatted differently than the input, the function will return false. The evaluation value must comply with the ARML formatting standards.

FIG. 17ZZ

Appendix "A"

- **MAXCHARS**, this function will take an input and evaluate whether or not the number of characters in the string is less than or equal to the evaluation value passed into the function. If the number of characters in the string is less than or equal to the evaluation value, true is returned. If the number of characters in the string is greater than the evaluation value, false is returned. All input values will be treated as a string data type.
- **MINCHARS**, this function will take an input and evaluate whether or not the number of characters in the string is greater than or equal to the evaluation value passed into the function. If the number of characters in the string is greater than or equal to the evaluation value, true is returned. If the number of characters in the string is less than the evaluation value, false is returned. All input values will be treated as a string data type.

Example:

```

<ACTION TYPE="IF">
  <COND EVAL="[QUERY1.STRREAD]" TYPE="EQUALS" VALUE="READ"></COND>
  <IFLIST>
    <ACTION TYPE="SAVE"></ACTION>
    <ACTION TYPE="OPEN" NAME="INBOX" NEWINST="FALSE"></ACTION>
  </IFLIST>
  <ELSELIST>
    <ACTION TYPE="OPEN" NAME="MSGREAD"
NEWINST="FALSE"></ACTION>
  </ELSELIST>
</ACTION>

```

FIG. 17AAA

Appendix "A"

Example of airix event model

The following example serves to illustrate how a screen is used to compose a data package to be sent back to the AIRIX server. The example used is a screen giving the bare functionality for composing a basic email message – to simplify the example, the user cannot cancel the action, and multiple recipients are not allowed.

```
<ARML>
  <SCREEN NAME="NewMsg">
    <BUTTONS>
      <BTN NAME="OK" CAPTION="Send" INDEX="0">
        <EVENTS>
          <EVENT TYPE="MODIFY">
            <ACTION TYPE="ARML">
              <ARMLTEXT>
                <BODY TYPE="ME">
                  <ME MSGID="1" FROM="Tim Neil"
                    SUBJECT="[SP.NewMsg.Subject]">
                    <DATA>[SP.NewMsg.Body]</DATA>
                    <RECIPS>
                      <RCP MSGID="1"
                        TO="[SP.NewMsg.To]"></RCP>
                    </RECIPS>
                  </ME>
                </BODY>
              </ARMLTEXT>
            </ACTION>
          </EVENT>
        </EVENTS>
      </BTN>
    </BUTTONS>
    <EDITBOXES>
      <EB NAME="To" INDEX="1"></EB>
      <EB NAME="Subject" INDEX="2"></EB>
      <EB NAME="Body" INDEX="3"></EB>
    </EDITBOXES>
  </SCREEN>
</ARML>
```

The Editboxes section at the bottom defines 3 editboxes, with the names of 'To', 'Subject', and 'Body';

```
<EB NAME="To" INDEX="1"></EB>
<EB NAME="Subject" INDEX="2"></EB>
<EB NAME="Body" INDEX="3"></EB>
```

FIG. 17BBB

Appendix "A"

There is one button on the screen, with the name of 'OK';

```
<BTN NAME="OK" CAPTION="Send" INDEX="0">
```

When the user clicks on OK, the button composes an ARML package to be sent to the AIRIX server;

```
<EVENT>
  <ACTION TYPE="ARML">
```

The ARML package sent is an 'ME' package as described in the example in section 4.2.1. It is composed as follows;

```
<BODY TYPE="ME">
  <ME MSGID="1" FROM="Tim Neil"
    SUBJECT="[SP.NewMsg.Subject]">
    <DATA>[SP.NewMsg.Body]</DATA>
    <RECIPS>
      <RCP MSGID="1" TO="[SP.NewMsg.To]"></RCP>
    </RECIPS>
  </ME>
</BODY>
```

The subject field is taken from the edit box named 'Subject';

```
<ME MSGID="1" FROM="Tim Neil" SUBJECT="[SP.NewMsg.Subject]">
```

The recipients field is taken from the edit box named 'Subject';

```
<RECIPS>
  <RCP MSGID="1" TO="[SP.NewMsg.To]"></RCP>
</RECIPS>
```

Finally the text of the message is filled from the 'Body' field;

```
<DATA>[SP.NewMsg.Body]</DATA>
```

FIG. 17CCC

Appendix “A”

7 AVM-server system interactions

This section describes the primitives that are used for system-level interactions that the AIRIX Smart Client has with the AIRIX server.

7.1 General

7.1.1 Description

System level packages are sent between AIRIX and the AVM (wirelessly).

7.1.2 Structure

System interactions are performed by exchanging ARML data packages with the following structure;

```
<ARML>
<HEAD>...</HEAD>
<SYS>
{data}
</SYS>
</ARML>
```

7.1.3 Tags

7.1.3.1 The <HEAD> tag

The package header is delimited by the <HEAD>...</HEAD> tags. Contained in text between the two tags is the id of the destination mobile. The HEAD tag has the following attributes;

Attribute	Optional?	Description
DT	No	The date & time in RFC 1123 format (including time zone)
ID	No	A unique ID for the message
VERSION	No	The version number of the application (currently “2.0”)
APPNAME	No	The application name (“0” for System Messages)
DEVICE	No	A numeric constant identifying the device
PID	Yes	A unique value used to designate a device.
AVMV	No	The version number of the Smart Client.

7.1.3.2 The <SYS> tag

The <SYS>...</SYS> pair contains the actual system package. The tag does not have any attributes.

FIG. 17DDD

Appendix "A"

7.2 Device Registration & deregistration package

7.2.1 Description

Device registration packages are sent from the AVM to the AIRIX server when a user registers their device.

7.2.2 Structure

A device registration package has the following structure;

```
{wrapper tags}
<REG>
    <USERNAME> {data} </USERNAME>
    <PASSWORD> {data} </PASSWORD>
</REG>
{wrapper tags}
```

7.2.3 Tags

7.2.3.1 The <REG> tag

The <REG>...</REG> pair delimit the registration request. The tag has no attributes.

7.2.3.2 The <USERNAME> tag

The <USERNAME>...</USERNAME> pair contain the user name. The tag does not have any attributes.

7.2.3.3 The <PASSWORD> tag

The <PASSWORD>...</PASSWORD> pair contain the password. The tag does not have any attributes.

7.2.4 Example

This package would be sent by a user, to register their device under a given name;

```
{wrapper tags}
<REG>
    <USERNAME>SUNTRESS</USERNAME>
    <PASSWORD>MYPASS</PASSWORD>
</REG>
{wrapper tags}
```

FIG. 17EEE

Appendix “A”

7.3 Registration confirmation package

7.3.1 Description

This package is sent back from the AIRIX server to the AVM to confirm that the device has been registered.

7.3.2 Structure

A registration confirmation package has the following structure;

```
{wrapper tags}
<REGCONFIRM>
  <VALUE> {data} </VALUE>
  <APPS>
    <APP></APP>
    <APP></APP>
  </APPS>
</REGCONFIRM>
{wrapper tags}
```

7.3.3 Tags

7.3.3.1 The <REGCONFIRM> tag

The <REGCONFIRM>...</REGCONFIRM> pair delimit the confirmation. The tag has no attributes.

7.3.3.2 The <VALUE> tag

The <VALUE>...</VALUE> pair contains the status of the registration request. The following text strings are allowable;

CONFIRM – this means that the registration request was successful

NOTREGPLATFORM – this means that the registration request failed because the device is not registered for the platform

INVALIDUSERPASS – this means that the registration request failed because the user name or password was not valid

NODEVICE – this means that the registration request failed because the device was not registered previously by an application

7.3.3.3 The <APPS> tag

The <APPS>...</APPS> pair contains a list of applications for the device.

FIG. 17FFF

Appendix "A"

7.3.3.4 The <APP> tag

The <APP>...</APP> pair contains an application header. It has the following attributes;

Attribute	Optional?	Description
ID	No	The application ID
NAME	No	The name of the application
DESCRIPTION	No	A text description of the application
REG	No	'YES' if the user is registered for this application. 'NO' if they are not.

7.3.4 Example

This package would be sent to confirm the example request in section 7.2.4;

```
{wrapper tags}
<REGCONFIRM>
  <VALUE>CONFIRM</VALUE>
  <APPS>
    <APP ID="4" NAME="EMAIL" DESCRIPTION="E-Mail Application"
    REG="YES">
    <APP ID="22" NAME="STOCKS" DESCRIPTION="Stock Quotes" REG="NO">
  </APPS>
</REGCONFIRM>
(wrapper tags)
```

FIG. 17GGG

Appendix “A”

7.4 Find applications package

7.4.1 Description

Find applications packages are sent from the AIRIX component to the AIRIX server when a user wishes to refresh their list of applications on a device

7.4.2 Structure

A device registration package has the following structure;

```
{wrapper tags}
<FINDAPPS>
</FINDAPPS>
{wrapper tags}
```

7.4.3 Tags

7.4.3.1 The <FINDAPPS> tag

The <FINDAPPS>...</FINDAPPS> pair delimit the application registration request. It has no attributes.

FIG. 17HHH

Appendix "A"

7.5 Find applications confirmation package

7.5.1 Description

This package is sent back from the AIRIX server to the AVM to and contains a list of applications available for the user

7.5.2 Structure

A registration confirmation package has the following structure;

```
{wrapper tags}
<FINDAPPSCONFIRM>
  <APPS>
    <APP></APP>
    <APP></APP>
  </APPS>
</FINDAPPSCONFIRM>
{wrapper tags}
```

7.5.3 Tags

7.5.3.1 The <FINDAPPSCONFIRM> tag

The <FINDAPPSCONFIRM>...</FINDAPPSCONFIRM> pair delimit the confirmation. The tag has no attributes.

7.5.3.2 The <APPS> tag

The <APPS>...</APPS> pair contains a list of applications for the device.

7.5.3.3 The <APP> tag

The <APP>...</APP> pair contains an application header. It has the following attributes;

Attribute	Optional?	Description
ID	No	The application ID
NAME	No	The name of the application
DESCRIPTION	No	A text description of the application
REG	No	'YES' if the user is registered for the application. 'NO' if they are not.

FIG. 17III

Appendix “A”

7.6 Application Registration & deregistration package

7.6.1 Description

Application registration packages are sent from the AIRIX component to the AIRIX server when a user wishes to register or deregister for an application.

7.6.2 Structure

A device registration package has the following structure;

```
{wrapper tags}
<APPREG>
</APPREG>
{wrapper tags}
```

7.6.3 Tags

7.6.3.1 The <APPREG> tag

The <APPREG>...</APPREG> pair delimit the application registration request. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	This defines the type of parameter. It can take two values; ADD – this means that the application is to be added to the registration database DELETE – this means that the application is to be removed to the registration database
ID	No	The ID of the application being registered/deregistered

FIG. 17JJJ

Appendix “A”

7.7 Application registration & deregistration confirmation package

7.7.1 Description

This package is sent back from the AIRIX server to the AVM to confirm that the application has been registered or deregistered.

7.7.2 Structure

A registration confirmation package has the following structure (note that for DELETE types, the <INTERFACE>...</INTERFACE> section will not be included);

```
{wrapper tags}
<APPREGCONFIRM>
  <INTERFACE>
    interface definition
  </INTERFACE>
</APPREGCONFIRM>
{wrapper tags}
```

7.7.3 Tags

7.7.3.1 The <APPREGCONFIRM> tag

The <APPREGCONFIRM>...</APPREGCONFIRM> pair delimit the confirmation. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	This defines the type of parameter. It can take two values; ADD – this means that the application is to be added to the registration database DELETE – this means that the application is to be removed to the registration database
ID	Yes	The ID of the application being returned (if any)

7.7.3.2 The <INTERFACE> tag

The <INTERFACE>...</INTERFACE> pair delimit the interface definition. The tag has no attributes, and contains an interface definition as laid out in section 3. Note that instead of the <DEVICES>...</DEVICES> tags in section 3.1.3.6, it will be replaced by <SCREENS>...</SCREENS> with the screen definitions for only the one device that the interface is being sent to (see section 3.4.3.2 for the definition of the

FIG. 17KKK

Appendix "A"

<SCREENS> tag). This section will **not be sent** for APPREGCONFIRM messages of TYPE="DELETE".

7.7.4 Example

The following example shows the application confirmation with screen definitions for an application that allows a user to view their inbox and the mails in it.

```
{wrapper tags}
<APPREGCONFIRM TYPE="ADD" ID="12">
  <INTERFACE>
    <AXSCHDEF>
      <EVENTS>
        <EVENT>
          (action definitions)
        </EVENT>
      </EVENTS>
      <AXTDEFS>
        (table definitions)
      </AXTDEFS>
      <DPACKETS>
        (data package definitions)
      </DPACKETS>
      <SCREENS>
        <SCREEN NAME="INBOX ">
          {screen definition}
        </SCREEN>
        <SCREEN NAME="VIEWNEWMAIL">
          {screen definition}
        </SCREEN>
      </SCREENS>
    </AXSCHDEF>
  </INTERFACE>
</APPREGCONFIRM>
{wrapper tags}
```

FIG. 17LLL

Appendix "A"

7.8 Setting the active device package

7.8.1 Description

If a user wishes to set the current device as their active device, the AVM must send a 'set active device' package to the AIRIX server

7.8.2 Structure

A 'set active device' package has the following structure;

```
{wrapper tags}
<SA>
{data}
</SA>
{wrapper tags}
```

7.8.3 Tags

7.8.3.1 The <SA> tag

The 'set active device' package is shown by the <SA>...</SA> tags. The tag has no attributes; the tag pair contains the user's username

7.8.4 Example

This package would be sent by a user with the username of 'scotty';

```
{wrapper tags}
<SA>scotty</SA>
{wrapper tags}
```

7.9 Set active device response

7.9.1 Description

This packages is sent back from the AIRIX server to the client in response to a request to set the current device as the active one.

7.9.2 Structure

A 'set active device response' package has the following structure;

```
{wrapper tags}
<SACONFIRM>
  <VALUE> {data} </VALUE>
</SACONFIRM>
```

FIG. 17MMM

Appendix "A"

{wrapper tags}

7.9.3 Tags

7.9.3.1 The <SACONFIRM> tag

The <SACONFIRM>...</SACONFIRM> pair delimit the confirmation. The tag does not have any attributes.

7.9.3.2 The <VALUE> tag

The <VALUE>...</VALUE> pair contains the status of the registration request. The following text strings are allowable;

CONFIRM – this means that the registration request was successful

NOTREGISTERED – this means that the registration request failed because

7.9.4 Example

This package would be sent by the AIRIX server to confirm a set active request;

```
{wrapper tags}
<SACONFIRM>
  <VALUE>CONFIRM</VALUE>
</SACONFIRM>
{wrapper tags}
```

7.10 Invalid Application package

7.10.1 Description

This package is sent back from the AIRIX server to the AVM in response to a request to interact with an application that is no longer registered with AIRIX.

7.10.2 Structure

An 'invalid application' package has the following structure;

```
{wrapper tags}
<NOAPP>
  <VALUE> {data} </VALUE>
</NOAPP>
{wrapper tags}
```

FIG. 17NNN

Appendix "A"

7.10.3 Tags

7.10.3.1 The <NOAPP> tag

The <NOAPP>...</NOAPP> pair delimit the confirmation. The tag has no attributes.

7.10.3.2 The <VALUE> tag

The <VALUE>...</VALUE> pair delimit the return code. It can only be NOAPPLICATION – Application not found.

7.10.4 Example

This package would be sent in response to a request if the application cannot be found;

```
{wrapper tags}
<NOAPP>
  <VALUE>NOAPPLICATION</VALUE>
</NOAPP>
{wrapper tags}
```

FIG. 17000

Appendix "A"

8 Application-server system interactions

The section that defines Application to server system interactions has been made obsolete by the document "AIRIX Polling XML Language Specification". It describes an XML-HTTP interface to AIRIX using POST and GET commands to a web-based ISAPI DLL.

FIG. 17PPP